Jarmo Harju
Geert Heijenk
Peter Langendörfer
Vasilios A. Siris (Eds.)

LNCS 5031

Wired/Wireless Internet Communications

6th International Conference, WWIC 2008
Tampere, Finland, May 2008
Proceedings

Springer

Jarmo Harju   Geert Heijenk
Peter Langendörfer   Vasilios A. Siris (Eds.)

Wired/Wireless Internet Communications

6th International Conference, WWIC 2008
Tampere, Finland, May 28-30, 2008
Proceedings

Springer

Volume Editors

Jarmo Harju
Tampere University of Technology
Department of Communications Engineering
P.O. Box 553, 33101 Tampere, Finland
E-mail: jarmo.harju@tut.fi

Geert Heijenk
University of Twente
Faculty of Electrical Engineering, Mathematics, and Computer Science
Chair of Design and Analysis of Communication Systems,
P.O. Box 217, 7500 AE Enschede, The Netherlands
E-mail: geert.heijenk@utwente.nl

Peter Langendörfer
IHP Microelectronics
Im Technologiepark 25, 15236 Frankfurt (Oder), Germany
E-mail: langendoerfer@ihp-microelectronics.com

Vasilios A. Siris
University of Crete, Computer Science Department
P.O. Box 2208, 71409 Heraklion, Crete, Greece
and
FORTH, Institute of Computer Science
P.O. Box 1385, 71110 Heraklion, Crete, Greece
E-mail: vsiris@ics.forth.gr

Library of Congress Control Number: 2008927239

CR Subject Classification (1998): C.2, D.4.4, D.2, H.3.5, H.4, K.6.4

LNCS Sublibrary: SL 5 – Computer Communication Networks and Telecommunications

ISSN	0302-9743
ISBN-10	3-540-68805-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-68805-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12277186 06/3180 5 4 3 2 1 0

Preface

WWIC 2008 was organized by the Technical University of Tampere, Finland, and it was the sixth event in a series of International Conferences on Wired/Wireless Internet Communications. Previous events were held in Coimbra (Portugal) in 2007, Berne (Switzerland) in 2006, Xanthi (Greece) in 2005, Frankfurt (Germany) in 2004, and Las Vegas (USA) in 2002.

The WWIC 2008 call for papers attracted 67 submissions from 33 countries, which were subject to thorough review work by the Technical Program Committee members and additional reviewers. The selection process resulted in acceptance of 18 papers, organized into 6 single-track technical sessions. The WWIC 2008 main technical program covered studies on performance analysis of wireless systems, topics on resource and QoS management, and issues on implementation techniques, mobility, cross-layer design and wireless sensor networks. The technical program was complemented by two keynote speeches, by Nestor Peccia (European Space Agency, Darmstadt, Germany) on "Interplanetary Internet," and Arto Karila (Helsinki Institute of Information Technology, Finland), on "Multi-Access in Regional Networks." In addition to the main technical program, the last day of the conference was dedicated to the Second ERCIM workshop on eMobility.

WWIC has been selected as the official annual conference by COST Action 290 (Wi-QoST-Traffic and QoS Management in Wireless Multimedia Networks) since 2005, and this time the conference was co-located with the final management committee meeting of the Action. For that reason, two sessions of WWIC were dedicated to the Action, and in those sessions a distinguished selection of the results of COST Action 290 were presented. Those papers were selected from the submissions to the general call for papers according to the same strict criteria as the other accepted papers.

We would like to thank the authors for choosing to submit their results to WWIC 2008. We would also like to thank all the members of the Technical Program Committee, as well as all the additional reviewers for their effort in providing detailed and constructive reviews. The support of Springer LNCS is gratefully acknowledged again this year. We are also grateful to the Organizing Committee for its efforts. We hope that all attendees enjoyed the scientific and social program as well as the city of Tampere and the surrounding lakes. We look forward to welcoming you at WWIC 2009, to be held in Twente (Enschede), The Netherlands.

May 2008

Jarmo Harju
Geert Heijenk
Peter Langendörfer
Vasilios Siris

Organization

Executive Committee

General Chairs	Jarmo Harju, Tampere University of Technology, Finland
	Peter Langendörfer, IHP Microelectronics, Germany
TPC Chairs	Vasilios Siris, University of Crete / FORTH-ICS, Greece
	Geert Heijenk, University of Twente, The Netherlands

Local Organizing Committee

Jakub Jakubiak	Tampere University of Technology, Finland
Sari Kinnari	Tampere University of Technology, Finland
Bilhanan Silverajan	Tampere University of Technology, Finland
Heikki Vatiainen	Tampere University of Technology, Finland
Kirsi Viitanen	Tampere University of Technology, Finland

Steering Committee

Torsten Braun	University of Bern, Switzerland
Georg Carle	University of Tübingen, Germany
Giovanni Giambene	University of Siena, Italy
Yevgeni Koucheryavy	Tampere University of Technology, Finland
Peter Langendörfer	IHP Microelectronics, Germany
Ibrahim Matta	Boston University, USA
Vassilis Tsaoussidis	Demokritos University, Greece
Nitin Vaidya	University of Illinois, USA

Supporting and Sponsoring Organizations

Project COST290
ERCIM
Nokia
European Science Foundation
Tampere University of Technology

Technical Program Committee

Ozgur B. Akan	Middle East Technical University, Turkey
Khalid Al-Begain	University of Glamorgan, UK
Manuel Alvarez-Campana	Universidad Politecnica de Madrid, Spain

Guevara Noubir	Northeastern University, USA
Evgeny Osipov	Luleå University of Technology, Sweden
Philippe Owezarski	LAAS-CNRS, France
George Pavlou	University College London, UK
Aleksi Penttinen	Helsinki University of Technology, Finland
George Polyzos	AUEB, Greece
Utz Roedig	Lancaster University, UK
Theodoros Salonidis	Thomson - Paris Research Labs, France
Guenter Schaefer	TU Ilmenau, Germany
Jochen Schiller	Free University Berlin, Germany
Arunabha Sen	Arizona State University, USA
Patrick Snac	ISAE, France
Dimitrios Serpanos	University of Patras, Greece
Vasilios Siris	University of Crete and ICS-FORTH, Greece
Dirk Staehle	University of Wuerzburg, Germany
Burkhard Stiller	University of Zurich and ETH Zurich, Switzerland
Phuoc Tran-Gia	University of Wuerzburg, Germany
Vassilis Tsaoussidis	Demokritos University, Greece
Thiemo Voigt	SICS, Sweden
Miki Yamamoto	Kansai University, Japan
Chi Zhang	Juniper Networks, USA
Martina Zitterbart	University of Karlsruhe, Germany

Additional Reviewers

Anwander, Markus
Basanta, Pablo
Bereketli, Alper
Bijwaard, Dennis
Cerqueira, Eduardo
Cheng, Wei
Chini, Paolo
De Cicco, Luca
Duarte, Ricardo
Dunaytsev, Roman
Ergul, Ozgur
Fischer, Mathias
Fonte, Alexandre
Frangoudis, Pantelis
Galera, Francisco
Giannetti, Samuele
Gligoroski, Danilo
Hadzic, Snezana
Horn, Werner
Isik, Mehmet
Kamateri, Eleni
Katsaros, Ntinos
Knapskog, Svein
Lacan, Jerome
Ma, Liran
Maeder, Andreas
Matos, Alexandre
Neves, Pedro
Oechsner, Simon
Oostveen, Job
Osterlind, Fredrik
Psaras, Ioannis
Shah, Ghalib
Tsioliaridou, Ageliki
Van Wambeke, Nicolas
Ververidis, Christopher
Wagenknecht, Gerald
Waldhorst, Oliver
Xing, Kai
Zhang, Haibin

Table of Contents

A New CAC Policy Based on Traffic Characterization in Cellular Networks

Natalia Vassileva and Francisco Barcelo-Arroyo

Universitat Politecnica de Catalunya (UPC),
c./ Jordi Girona 1-3, Campus Nord, C3
08034, Barcelona, Spain
{natalia,barcelo}@entel.upc.edu

Abstract. The Call Admission Control (CAC) method presented in this paper is based on the statistical properties of the network's traffic variables. It probabilistically estimates the time until the release of a seized channel: the admission control depends on the computed mean remaining time averaged along all channels at a specific instant and on a time threshold. The policy produces a smooth transition between the QoS metrics, giving the operator the freedom to design the network at the desired QoS point. Another valuable property is that the algorithm is straightforward and fed only by simple teletraffic metrics: distribution and the first and second moments of Channel Holding Time (CHT). Simplicity is important for a CAC method because decisions for accepting or rejecting calls must be computed quickly and frequently.

Keywords: QoS parameters, call admission control (CAC), resource allocation schemes, traffic engineering, wireless cellular systems.

1 Introduction

With the increasing number of users and demand for more services in cellular systems, reducing the size of cells is one of the measures typically undertaken to increase the traffic capacity. Smaller cell size leads to more Handovers (HO), and therefore efficient resource allocation mechanisms are required to guarantee the continuation of ongoing calls when Mobile Stations (MS) cross cell boundaries (switch from one Base Station (BS) to another). HO schemes are usually evaluated through two Quality of Service (QoS) parameters: Probability of Blocking *(PB*) of a new call and Probability of Failure (*PF*) of handover calls. The former metric evaluates the probability of denying service at the beginning, when a call attempt is received; the latter represents the probability of interrupting a call in progress due to handover drop. In order to bring wireless cellular systems closer to the fixed ones, CAC strategies attempt to reduce *PF* while maintaining *PB* at an acceptable level. The Probability of Dropping (*PD*) a handover is a parameter of interest to the operator, but the user perceives only *PF* (see Section 4).

This work introduces a new concept into HO schemes – the implementation of a CAC based on the statistical properties of a network's traffic variables. The admission control presented here considers the Channel Holding Time (CHT). CHT comprises the time from the instant the channel is assigned to a call (new or handover) until the instant the channel is released. In other words, it is the time spent by the MS in the same cell (i.e., connected to the same BS) while talking. Because of the mobile nature of the MSs, CHT generally differs from the unencumbered call duration. In this work, it is used to probabilistically determine the remaining holding time of the busy channels at a specific instant. This metric can then be used in the design of CAC strategies.

Simplicity is important for a CAC method because the conditions for accepting or rejecting a call must be computed quickly and frequently. Another valuable property is the ability to find the desired balance between *PB* and *PF* in order to allow the operator to design the network at the preferred QoS point. As shown below, the CAC introduced in this work is straightforward to compute, is fed only by simple teletraffic metrics, and produces a smooth transition between *PB* and *PF*.

1.1 Handover Schemes

CAC methods prioritize handover traffic before fresh call attempts – HO calls are always admitted as long as there are available resources. Various strategies are used for that purpose. A classic one is the Guard Channel Scheme (GCH) [1], [2], which uses a cutoff policy to reserve a number of resources to be available only to calls in progress. This scheme can include the possibility of a handover to seize a guard channel first, a common channel first, or a probabilistic combinations of both [3]. GCH schemes have only one parameter (the number of guard channels) to tune, and this parameter is an integer value. This makes it difficult to find the desired balance between *PB* and *PF*. In addition, a general drawback of the cut-off policies is that their use of resources is far from optimal. The Handover Queueing Scheme (HQS) [4] gives priority to HO calls by permitting them to queue. The attained Carried Traffic (CT) is good, but in general handover calls are too highly prioritized. An extension to these schemes is the Guard Channel with Queue (GCQ) scheme, in which new calls have access to only part of the total capacity and cannot queue.

Fractional Guard Channel (FGC) [1] both exercises finer control of *PB* and *PF* probabilities and achieves higher carried traffic than GCS. The Dynamic Guard Channel (DGC) scheme [5] uses the mobility and number of busy channels heuristically in order to allocate free resources to incoming new calls. Other handover priority strategies implement measurement-based parameters, such as transmitted power, time spent in the degradation area, and traffic load in neighbouring cells [3], [4], [6], and [7]. These schemes permit the easy transition between *PB* and *PF*, but the number of parameters to adjust is so high that these policies are difficult to tune; tuning too many parameters has a high computational cost and makes these schemes impractical [8], [9].

1.2 Goal and Organization

During analysis of handover schemes, traffic variables are usually assumed to be exponentially distributed to obviate the analytical intractability of the problem. Empirical evaluations [10], [11], [12], and [13] demonstrate that most of the traffic processes that take place in the cellular systems differ from Poisson or negative exponential distribution (n.e.d.). For example, HO arrivals, CHT, dwell time in the overlap area, and other traffic-related random variables (r.v.) are generally not memoryless.

The main goal of this work is to study the applicability of statistical knowledge of the processes observed in cellular systems to the control of QoS parameters. The methodology applied is as follows. The common hypotheses about the holding time are relaxed. At the same time, the assumptions usually made about the arrival process are still present. This allows us to isolate the implications of holding time for CAC functionality and system performance from the implications of the arrival process. The method presented here uses the duration of channel occupancy. The hypothesis for a n.e.d. CHT is relaxed. Based on the time elapsed from seizing a free channel, the remaining time for releasing it can be probabilistically estimated. The estimation of the remaining CHT can be used to compute the mean remaining time for freeing a channel averaged along all the channels. By knowing its value and setting a restrictive or looser time threshold, a different priority can be given to HO traffic. This provides a window of possible values from which the operator can choose according to the desired trade-off between QoS and carried traffic.

The paper is organized as follows. In Section 2, the analysis of the remaining CHT is reviewed. In Section 3, the HO method is presented. Section 4 outlines the simulation setup, and Section 5 contains the performance figures. Section 5 concludes the paper with summary remarks and directions for future work.

2 Expected Remaining Time

The CAC method described in Section 3 implements a metric related to the elapsed channel holding time – the remaining time of a call in service in a channel. We use the terminology introduced in [11] to designate by "remaining time" the time interval between the instant when a decision for a new call acceptance/rejection has to be made and the instant when a resource will become free (i.e., call termination or continuation of the call in a new cell).

The mathematical analysis elaborated and notation used here closely follow those of [11] and [14]; for this reason, only their main results are provided here. According to [14], the remaining time can be estimated by applying the following analysis. It is assumed that a new call can arrive at any instant with equal probability. If *h* is the remaining time of an ongoing call with a cumulative distribution function (cdf) $F(t)$ and mean service time m_1, then *h* is distributed according to the following probability distribution function (pdf):

$$f_h(t) = \frac{1 - F(t)}{m_1} \ . \tag{1}$$

The interest is focused on determining how long the seized resource will remain busy (i.e., in the distribution of the remaining CHT time of an ongoing call) given that the service has already been in progress during time ε. The elapsed time ε from the beginning of a service is known by the network and can be used to calculate the conditional density of $h, f_h(t, \varepsilon,)$, in the following way:

$$f_h(t,\varepsilon) = \frac{f(t+\varepsilon)}{1 - F(\varepsilon)} = \frac{f(t+\varepsilon)}{1 - \int_0^{\varepsilon} f(t)dt} \ . \tag{2}$$

The average remaining time $\bar{h}(\varepsilon)$ then can be computed from that pdf as:

$$\bar{h}(\varepsilon) = \int_0^\infty t f_h(t,\varepsilon) dt \ . \tag{3}$$

The hypothesis for exponentially distributed CHT was relaxed to permit a holding time with a Square Coefficient of Variation (SCV) different from one. Here, the case of Hyper-Exponential-2 (HE-2) distributed holding time is studied. Some system models differentiate between connections belonging to calls that remain in the same cell and those that require a handover [15], and [16]; others consider two different types of connections (e.g., voice and data). Both cases can be modelled by HE-2 distributions (i.e., by the combination of two n.e.d. r.v. with different means). For simplicity and without loss of generality, a balanced HE-2, in which the time consumed by the two types of n.e.d. combined in the HE-2 is the same, is used in this paper. Erlang-3-distributed CHT (SCV = 1/3) is also used in order to study the system for the case of a SCV lower than one. In [10], the Erlang-k distribution is used to fit empirical data for message holding time in cellular systems.

Table 1. Pdf of remaining time (*$f_h(t)$*), pdf of remaining versus elapsed time (*$f_h(t,\varepsilon)$*), and average remaining time (*$h(\varepsilon)$*) for Hyper-exponential-2 distributed CHT

$f_h(t) = \frac{1}{m_1}(pe^{-\mu_1 t} + (1-p)e^{-\mu_2 t})$	(4)
$f_h(t,\varepsilon) = \frac{p\mu_1 e^{-\mu_1(t+\varepsilon)} + (1-p)\mu_2 e^{-\mu_2(t+\varepsilon)}}{pe^{-\mu_1\varepsilon} + (1-p)e^{-\mu_2\varepsilon}}$	(5)
$\bar{h}(\varepsilon) = \frac{1}{\mu_1}\frac{pe^{-\mu_1\varepsilon}}{pe^{-\mu_1\varepsilon} + (1-p)e^{-\mu_2\varepsilon}} + \frac{1}{\mu_2}\frac{(1-p)e^{-\mu_2\varepsilon}}{pe^{-\mu_1\varepsilon} + (1-p)e^{-\mu_2\varepsilon}}$	(6)

The algorithm is not limited to specific scenarios. The aforementioned distributions are used to exemplify the principal idea and functionality of the presented CAC method. Concrete distributions for a given system with a different SCV should lead to similar qualitative but not quantitative results with a gradual change in the QoS metrics. In Table 1 and Table 2, the main analytical results reviewed are summarized for the used distributions.

For the HE-2-distributed CHT the average remaining time, $\bar{h}(\varepsilon)$, is a monotonically increasing function. The longer the elapsed time, the longer the average remaining. In Table 1, Eq. 4, m_1 is the mean duration of the HE-2 distribution (i.e. $m_1 = 1/\mu = p/\mu_1 + (1-p)/\mu_2$).

In contrast to the HE-2, the average remaining time for the Erlang-3 distribution is a decreasing function. The longer the elapsed time of a call during its course, the higher the probability that the channel will soon be released.

Table 2. Pdf of remaining time ($f_h(t)$), pdf of remaining versus elapsed time ($f_h(t,\varepsilon)$), and average remaining time ($h(\varepsilon)$) for Erlang-3

$f_h(t) = \frac{\mu}{3} e^{-\mu t}(1 + \mu t + \frac{(\mu t)^2}{2})$	(7)
$f_h(t,\varepsilon) = \frac{1}{2} \frac{\mu^3 (t+\varepsilon)^2 e^{-\mu t}}{1 + \mu\varepsilon + \frac{(\mu\varepsilon)^2}{2}}$	(8)
$\bar{h}(\varepsilon) = \frac{6 + 4\mu\varepsilon + (\mu\varepsilon)^2}{2\mu + 2\mu^2\varepsilon + \mu^3\varepsilon^2}$	(9)

3 System Study with Mean Remaining Time Policy

The CAC outlined uses simple traffic metrics, such as average CHT, SCV, and number of free/busy channels, as inputs. The latter are readily available, and the first two are easy to compute on-line in the network. They can be estimated along different time windows that must be long enough to allow averaging and short enough to assume that the traffic processes are stationary. When the normal duration of connections and ITU-T recommendations on teletraffic are considered, this window is typically one hour. The distribution of the CHT is assumed to be known in the network. Operating knowing only the first two moments, without a precise knowledge of the distribution, would also be possible, but this task is left for further study.

The algorithm relies on the probabilistically defined remaining CHT of the busy channels. The average remaining time of each busy channel is estimated according to Eq. 3. It is set to zero ($\bar{h}(\varepsilon) = 0$) for the free channels in order to account for currently available resources. A HO call is served as long as there are free channels. In order to determine whether a new call is to be admitted or rejected, the Mean Remaining Time (*MRT*) is averaged along all the channels (i.e., busy and idle; denoted with *C* in Fig. 1 and formulae) at a specific instant of time, computed as follows:

$$MRT = \frac{1}{C} \sum_{i=1}^{C} \bar{h}_i(\varepsilon) \ . \tag{10}$$

The ongoing calls are prioritized by setting a Time Threshold (*TT*). The estimated mean remaining time to release a channel is compared to the time threshold. If a free channel is available and *MRT*<*TT*, the new call will be admitted to seize it and will not increase the probability of a call interruption (*PF*), according to the estimated parameter (see Fig. 1). If *MRT*>*TT,* however, the new call will be rejected in order to admit future HO arrivals.

```
/*Guard Channel Scheme (GCS)*/
    // GC:  Guard Channels
    // C: Capacity
    // BC: Busy Channels

    if (BC < (C – GC))
            then accept call;
    else
            if new call
                    then block call;
            else /* HO call */
                    if  (BC < C)
                        then accept call;
                    else
                        drop call;
```

```
/*Mean Remaining Time (MRT)*/
    // TT: Time Threshold
    // C: Capacity
    // BC: Busy Channels

    if (BC < C)
            if new call
                    if (MRT < TT)
                            then accept;
                    else
                            block call;
            else   /* if HO call */
                    accept call;
    else   /*BC=C, i.e. all channels busy*/
            block/drop call;
```

Fig. 1. Guard Channel Scheme (GCS) and Mean Remaining Time (MRT) Scheme

The GCS can be seen as a particular case of the MRT algorithm for exponentially distributed CHT. Since the estimated residual life of a n.e.d. random variable is equal to its average value independent of the instant of observation [14], the estimation of the remaining CHT does not depend on the elapsed time. All the Busy Channels (*BC*) will have the same remaining time independent of the elapsed time ε in each busy channel j:

$$\bar{h}_j(\varepsilon) = \bar{h} = \frac{1}{\mu} \ , \tag{11}$$

where $1/\mu$ is the average CHT $(m_1 = 1/\mu)$. Independent of the instant of time when *MRT* is computed, its value will only depend on the number of *BC*, i.e.:

$$MRT = \frac{1}{C} \cdot \frac{BC}{\mu} \ . \tag{12}$$

The correspondence between the two schemes when the CHT is n.e.d. is more clearly seen if the time threshold is set equal to:

$$TT = \frac{1}{\mu} \cdot \frac{C - GC}{C} \ , \tag{13}$$

where *GC* stands for Guard Channels as in Fig. 1. Then, the condition *MRT<TT* is reduced to that of the GCS (Fig. 1): *BC<C–GC* (*C–GC* is the common pull of channels available to new and HO calls). Indeed, it is straightforward that, for a particular value of *MRT* and $TT \in (x-1; x)$, where x is a positive integer number, the system response to the admission/rejection of new calls will remain the same and can only change when *TT* undergoes integer changes. A comparison of the two methods, along with the subtle difference between them, is included in the performance part of the paper.

As analytically demonstrated in Section 2, the elapsed time probabilistically determines the remaining time to release a channel. For the HE-2-distributed CHT, the longer the elapsed time of an ongoing call in a given BS, the longer the expected time it will remain in service (occupying the assigned resource). Thus, the greater the number of channels with an estimated average *h(ε)* of high value, the greater the *MRT* will be. The value of *MRT* for a given time threshold defines the acceptance or rejection of a new call. If the time threshold is restrictive, the *PB* for new calls for a particular *TT* will be elevated for most of the interval of the estimated *MRT* values. This is the intended logic behind the algorithm: if the state of the busy channels stays unchanged for a long period of time, no more "fresh" calls are to be accepted; otherwise, HO calls will be rejected and *PF* will rise correspondingly. If many channels are free and/or the estimated *MRT* is low, however, "fresh" call requests are to be admitted in the BS to thereby improve *PB* and efficiently use the system capacity. Bearing in mind the relationship between the elapsed and remaining time (i.e., the longer ε, the smaller the average *h(ε)* will be), reciprocal reasoning can be applied to Erlang-3.

4 Simulation Setup

In order to obtain the performance metrics of interest, Omnet++ [17] was used to simulate the MRT scheme. Omnet is a modular discrete event network simulator. The teletraffic system was implemented by designing the following modules: Traffic generator (for generating calls – new and HO with a user-defined distribution), Dispatcher (executes the logic of the scheme), Server (for each of the channels), and Statistics (collects the output of the simulation runs; see next subsection on metrics computed and stored in this module). In order to validate the simulator, first the teletraffic system without a HO policy (all calls are accepted as far as there are available resources) and the GCS method were simulated. For these systems, analytical results can be obtained through Markov chains. An excellent agreement between the analytical and simulated results was observed. A similar approach and methodology for studying the performance of traditional HO schemes is applied in [18].

Several hypotheses common to other CAC studies (see [19] for example) are assumed in this paper. The wireless system under study is homogenous, and so is the traffic. All cells have the same size and capacity and work under identical traffic conditions (i.e., the arrival intensities of new and HO calls). The system offers only voice service. As a result, it is sufficient to model and simulate the performance of one cell.

In order to isolate the implications of non-Poisson CHT, the arrival process is assumed to be Poisson with an arrival intensity λ. This describes Poisson "fresh" and handover incoming traffic with rates λ_{new} and λ_{HO}, respectively. These metrics are interrelated by the following equations:

$$\lambda = \lambda_{new} + \lambda_{HO}, \qquad \alpha = \frac{\lambda_{HO}}{\lambda_{new}} , \tag{13}$$

where α is the mobility factor, which is the ratio of handover to new call arrival rates to the BS (an estimation of mobility). In this paper reported average values for α and the unencumbered call duration (d_{call}) are applied, the average duration of the channel holding time ($1/\mu$ in the formulae and Table 3) can be defined in the following way:

$$\frac{1}{\mu} = \frac{d_{call}}{\alpha + 1} \quad . \tag{14}$$

Thus the average CHT is equal to the average call duration divided by the average number of handovers per call plus one; this is the average number of visited cells.

The metrics that the simulation program gives as output values are the number of blocked and served calls. The following formulae are used to compute the performance metrics of interest:

$$PB = \frac{Number\ New\ Blocked}{Number\ New\ Blocked + Number\ New\ Served} \ , \tag{15}$$

$$PD = \frac{Number\ HO\ Blocked}{Number\ HO\ Blocked + Number\ HO\ Served} \ , \tag{16}$$

where *PD* is the Probability of Dropping a HO call. *PD* describes the probability of rejecting a HO request, whereas *PF* is the probability that one of the HO attempts that a call will require along its duration will be dropped and thus will lead to a forced termination of the ongoing call. Since *PD* cannot be perceived by the users, *PF* is computed through it. If we take into account that a call on average visits α+1 cells (i.e., requires α HO), the probability of failure can be computed by [22]:

$$PF = 1 - (1 - PD)^{\alpha} \ . \tag{17}$$

The input parameters and their feeding values are summarized in Table 3. The results shown next are obtained for a mobility of α=2. It is in accordance with the reported value in [10]. Note that the average CHT time is defined using Eq. 13 with a reported value for the whole call duration of 120 seconds [10].

Table 3. Simulated scenarios

A	C	α	1/μ	SCV	TT
4.5 and 6.5Erl	10	2	40 s.	10 and 1/3	0 – ∞

5 Performance Figures

The performance results in Fig. 2 for light (4.5 Erl – 45% loaded) and medium (6.5 Erl – 65% loaded) traffic for the two different CHT distributions demonstrate that the MRT policy smoothly controls the values of the probability of blocking and failure as a function of the time threshold. Since *PB* and *PF* are interrelated, the changes they experience are expected. Increasing the *TT* allows more new calls to enter the system (*PB* decreases), whereas it causes HO calls to become less prioritized (*PF* increases).

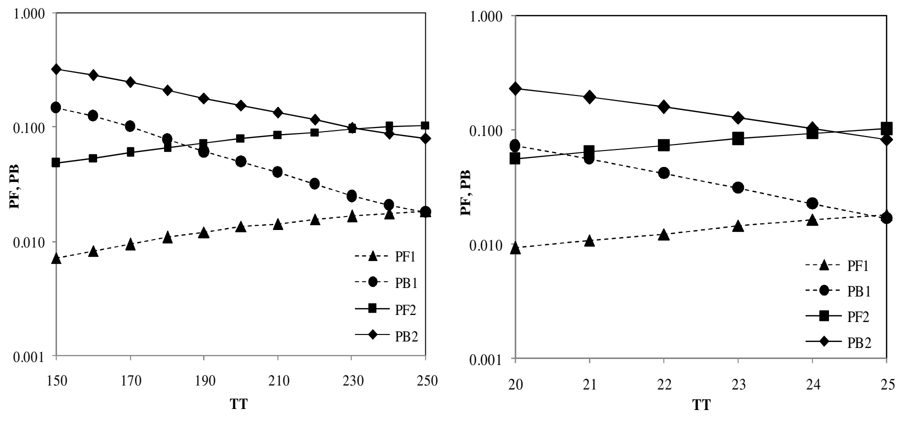


Fig. 2. QoS performance figures for 1) light (45%) and 2) medium (65%) traffic load. On the left, HE-2-distributed CHT is shown; on the right, Erlang-3 distributed CHT is shown.

Various simulations were executed for the whole range of possible values of the time threshold under different loads. In Fig. 2, of all the possible values of *PF* and *PB*, only the interval of practical interest is presented. It is limited by the blocking probability, which is regarded by operators as unacceptable when exceeding 20%. Above a certain point (depending on the distributions and particular values of the CHT: *TT*>30 and *TT*>400 for Erlang-3 and HE-2 in these scenarios), the *TT* is so loose that all the traffic is accepted when there are free channels.

Table 4 stores the analytical figures of *PF* and *PB* for GCS in order to allow comparison with MRT under the same traffic scenarios. Pure loss (i.e., queueless) teletraffic systems are in general insensitive to the distribution of the holding time [20], and GCS in particular is insensitive to the CHT distribution [21]. Note that the first three cases are of practical significance (*PB*<20%) and are QoS points that can be accomplished by the MRT method as well. Therefore, for GCS, the choice of the operator is very limited and restricted to two or three (depending on the traffic load) possible working points. In contrast, it can be observed from the figures that the MRT offers a continuous interval of possible *PB* and *PF* pairs. When the load offered to the system increases, the working interval becomes tighter for both schemes. With the MRT, however, there is still a wider operational window compared to the traditional cut-off method. The gradual transition of the blocking and failure probabilities is important, because it gives the operator the freedom to finely adjust the *PF* and *PB* figures.

Table 4. Performance figures for the Guard Channel Scheme with 1) 45% and 2) 65% traffic load, capacity $C = 10$ channels, and $\alpha = 2$

Guard Channels	PB_1	PF_1	PB_2	PF_2
1	0.030	0.014	0.135	0.081
2	0.068	0.010	0.230	0.057
3	0.132	0.007	0.347	0.041
4	0.232	0.005	0.483	0.031

Various performance goals can be set and achieved through the presented CAC method. One possible target value is included in Table 5 along with the corresponding *TT* value. It shows an interesting case, in which the probability of failure is targeted to be five times smaller than the blocking probability, *PB* is maintained within reasonable limits (e.g., less then 6% for HE-2 and around 5.6% for Erlang-3- distributed CHT), and the carried traffic is 4.4Erl of the attainable 4.45Erl when no prioritization scheme is implemented.

Table 5. Performance objectives for light load (*A*=4.5 Erl) and corresponding *TT*

CHT	*TT* for (5x*PF*=*PB*)	*PB* (%)	*PF* (%)
HE-2	190	6.12	1.21
Erlang-3	21	5.62	1.08

The Carried Traffic (*CT*) is a measure of the efficiency of system capacity utilization and is of interest to network operators. It is calculated using the output of the simulation in the following way [22]:

$$CT = A(1 - \frac{PB + PF}{\alpha + 1}) \ . \tag{18}$$

With the decrease of *PB* (increase of *PF*), the channel efficiency utilization is increased, since more new calls are admitted and more traffic is carried. The augmentation of the *CT* increases the operator's revenue, and thus performance figures are usually adjusted according to both the target QoS and *CT* values – an acceptable tradeoff between quality of service and revenue is sought. In the light of this, Fig. 3 shows the *CT* for the MRT scheme for medium load. It also displays the carried traffic attained by GCS: 6.17 Erl, 5.98 Erl, and 5.73 Erl for the three working points (i.e., for 1, 2, and 3 *GC*, Table 4). The MRT scheme performs equally well under the HE-2

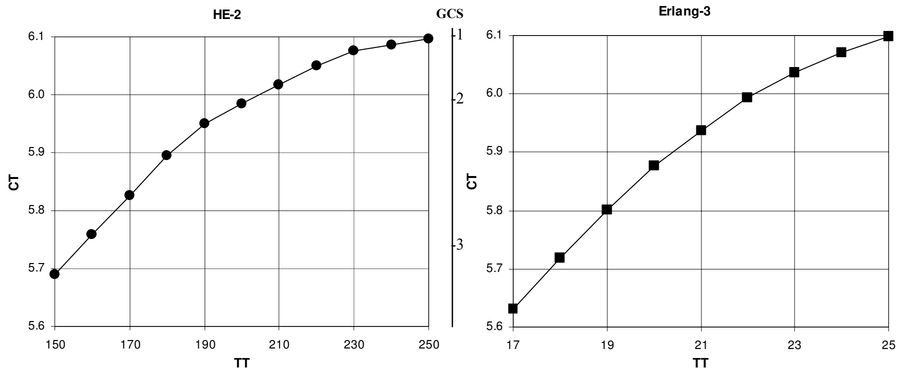


Fig. 3. Carried traffic for a medium (65%) traffic load using the MRT and GCS method

and Erlang-3 distributions. Compared to the cut-off scheme with regard to the carried traffic, there are clearly no gains in terms of *CT*. Benefits do exist, however, for the range of possible QoS pairs and *CT* from which the operator can choose. It can be observed that the operational interval offered by MRT is continuous, which is the main advantage of the algorithm.

6 Conclusion

In this work, the application of the expected remaining service time to the design of CAC algorithms was studied and its performance examined via simulation. The results obtained demonstrate that the implementation of CAC with statistical knowledge is advantageous to smoothly regulating the balance between different QoS metrics while still maintaining a high level of simplicity and ease for quick decision computation. When compared to traditional policies like GCS, the MRT method facilitates a wider operational interval. This interval gives the operator the freedom to choose the desired trade-off between QoS and revenue.

This is the first study of CAC with traffic characterization. Further research is required for designing algorithms for scenarios in which the carried traffic is higher but the desired QoS is still met. Also, motivated by the results obtained when relaxing the hypothesis for the holding time distribution, future work will investigate the implications of the arrival process on the design of CAC methods.

Acknowledgments. This research was funded by the Spanish Government and FEDER through the Plan Nacional de I+D (TEC2006-09466/TCM).

References

1. Ramjee, R., Nagarajan, R., Towsley, D.: On optimal call admission control in cellular networks. In: IEEE INFOCOM, pp. 45–50 (1996)
2. Yoon, C.H., Un, C.K.: Performance of personal portable radio telephone systems with and without guard channels. IEEE J. Selected Areas Communications 11, 911–917 (1993)
3. Kulavaratharasha, M.D., Aghvami, A.H.: Teletraffic performance evaluation of microcellular personal communication network (PCNs) with prioritized hand-off procedures. IEEE Trans. Vehicular Technology 48, 137–152 (1999)
4. Tekinay, S., Jabbari, B.: Handover and channel assignment in mobile cellular networks. IEEE Communications Magazine 29, 42–46 (1991)
5. Kim, Y.C., Lee, D.E., Lee, B.J., Kim, Y.H., Mukherjee, B.: Dynamic channel reservation based on mobility in wireless ATM networks. IEEE Communications Magazine 37, 47–51 (1999)
6. Ramanathan, P., Sivalingam, K.M., Agrawal, P., Kishore, S.: Dynamic resource allocation schemes during hand-off for mobile multimedia wireless networks. IEEE J. Selected Areas Communications 17, 1270–1283 (1999)
7. Agrawal, P., Ankevar, D.K., Narendran, B.: Channel management policies for handovers in celular networks. Bell Labs Technical Journal, 97–110 (1996)
8. Bisaws, S.K., Sengupta, B.: Call admissibility for multirate traffic in wireless ATM networks. In: IEEE INFOCOM, pp. 649–657 (1997)

9. Garcia, D., Martinez, J., Pla, V.: Comparative evaluation of admission control policies in cellular multiservice networks. In: Int. Conf. Wireless Communications, pp. 517–531 (2004)
10. Barcelo, F., Jordan, J.: Channel holding time distribution in public telephony systems (PAMR and PCS). IEEE Trans. Vehicular Technology 49, 1615–1625 (2000)
11. Barcelo, F.: Statistical properties of silence gap in public mobile telephony channels with application to data transmission. In: IEEE Int. Conf. Communications (ICC), pp. 2011–2015 (2001)
12. Chlebus, E.: Empirical validation of call holding time distribution in cellular communications systems. In: Proc. 15th Int. Teletraffic Congress (ITC), pp. 117–1189 (1997)
13. Jedrzycki, C., Leung, V.C.M.: Probability Distribution of Channel Holding Time in Cellular Telephone Systems. In: IEEE Vehicular Technology Conf (VTC), pp. 247–251 (1996)
14. Kleinrock, L.: Queueing systems. Theory, vol. I. John Wiley & Sons, Chichester (1975)
15. Chih-lin, I., Greenstein, J.L., Gitlin, R.D.: A Microcell/macrocell cellular architecture for low- and high-mobility wireless users. IEEE J. Selected Areas Communications 11, 885–891 (1993)
16. Steele, R., Nofal, M.: Teletraffic performance of city street microcells catering for pedestrian mobile users. In: IEE Colloquium on Univ. Research in Mobile Radio (1990)
17. Omnet++ Communite Site, `http://www.omnetpp.org`
18. Xhafa, A.E., Tonguz, O.K.: Handover performance of priority schemes in cellular networks. IEEE Trans. Vehicular Technology 57, 565–577 (2008)
19. Hong, D., Rappaport, S.S.: Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized hand-off procedures. IEEE Trans. Vehicular Technology VT-35, 77–92 (1986)
20. Iversen, V.: Handbook in Teletraffic Engineering. ITC/ITU-D (2005)
21. Xhafa, A.E., Tonguz, O.K.: Does mixed lognormal channel holding time affect the handover performance of guard channel scheme? In: IEEE GLOBECOM, vol. 6, pp. 3452–3456 (2003)
22. Barcelo, F.: Performance analysis of handoff resource allocation strategies through state-dependent rejection scheme. IEEE Trans. on Wireless Communications 3, 900–909 (2004)

Cross-Layer Modeling of TCP SACK Performance over Wireless Channels with Completely Reliable ARQ/FEC

Dmitri Moltchanov, Roman Dunaytsev, and Yevgeni Koucheryavy

Department of Communications Engineering, Tampere University of Technology
P.O. Box 553, FIN-33101, Tampere, Finland
{moltchan,dunaytse,yk}@cs.tut.fi

Abstract. We propose an analytical model for a TCP SACK connection running over a wireless channel with completely reliable ARQ/FEC. We develop the model in two steps. At the first step, we consider the service process of the wireless channel and derive the probability distribution function of the time required to successfully transmit a single IP packet over the wireless channel. This distribution is used at the next step of the modeling where we derive the expression for TCP SACK steady state goodput. The developed model allows to quantify the effect of many implementation-specific parameters on TCP performance in wireless domain. We also demonstrate that TCP spurious timeouts, reported in many empirical studies, do not occur when wireless channel conditions are stationary and their presence in empirical measurements should be attributed to non-stationary behavior of wireless channel characteristics.

1 Introduction

Nowadays, about 80-90% of all packets and bytes traversed over the Internet are TCP traffic and there is no indication that these numbers may decline in the future. Predicting TCP behavior in various environments is crucial for better understanding and optimizing TCP performance over modern networks. TCP congestion and flow control mechanisms pose significant challenges for a performance modeler. Given additional complexity of local error concealment mechanisms implemented at wireless channels previous works on TCP performance in wireless domain were mainly limited to simulation and measurement studies (e.g., see [1] and references therein). While empirical studies are extremely useful in TCP performance evaluation, it is a difficult task to simulate and explore TCP behavior across the range of all possible operational conditions and low-level protocols settings. In this case, an analytical cross-layer model is extremely beneficial because it allows to study the protocol performance over the entire parameter space and very easily apply the "what if" test to the different scenarios.

Due to negligibly low error ratio of current wired networks, performance degradation mainly stems from buffering procedures. The situation is much more complicated in wireless networks, where, in addition to performance degradation induced by buffering procedures, we also have to take into account those delays and losses occurring due to unreliable nature of wireless transmission media. Being inherently prone to transmission errors, wireless access technologies implement error concealment techniques including automatic repeat request (ARQ) and forward error correction (FEC). However, even in the presence of these local error recovery mechanisms bit errors may still propagate to higher layers resulting in loss or excessive delay of IP packets triggering TCP congestion control procedures. Therefore, the choice of local error concealment parameters may severely affect TCP performance.

In this paper, we develop a cross-layer model for a TCP SACK connection running over a wireless channel. The performance parameter of interest is the long-term steady state goodput of a single TCP SACK connection. The wireless channel characteristics are assumed to be stationary and modeled by a homogenous Markov process. We also assume that ARQ and FEC are both implemented at the wireless channel. The proposed model highlights many interesting features of TCP performance in wireless environment. Among other conclusions, we demonstrate that TCP spurious timeouts, reported in many empirical studies, do not occur when wireless channel conditions are stationary and should be attributed to non-stationary behavior of wireless channel characteristics.

The rest of the paper is organized as follows. The system model is described in Section 2. The cross-layer model is introduced in Sections 3 and 4. Numerical results are discussed in Section 5. Finally, conclusions are summarized in Section 6.

2 System Model

The system of interest is illustrated in Fig. 1. We consider a TCP connection between two hosts such that the last link on the end-to-end path from the sender to the receiver is a wireless channel. In this paper, we consider the wireless channel as a bottleneck. Since such scenario is common in wireless communications, we do not resort to a particular wireless access technology. Instead, we consider a number of features this channel includes. We assume that ARQ and FEC functionalities are both implemented at the wireless channel and ARQ operates according to either stop-and-wait or selective acknowledgement regime.

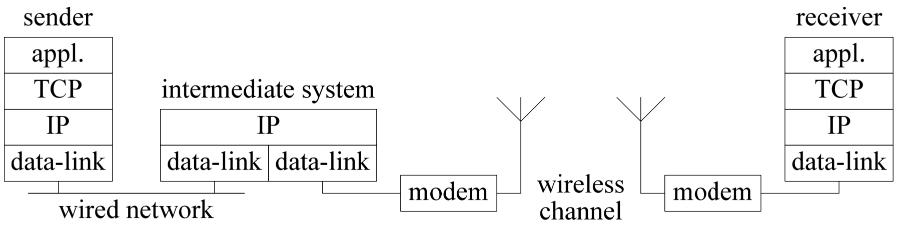


Fig. 1. System model

Since nearly all TCP implementations now support the selective acknowledgement (SACK) option, we assume that the sender and the receiver are both SACK-capable and use the SACK option under all permitted circumstances in accordance to [2], [3], and [4]. We consider an application process which always has data to send, thus the sender always sends full-sized TCP segments (i.e., containing *MSS* bits of data) whenever congestion window (*cwnd*) allows. We assume that the receiver's buffer is sufficiently large, so TCP send rate is not limited by inappropriately small value of the advertised receive window. We also assume that the time needed to send a window of segments is smaller than the round-trip time (*RTT*). These assumptions are justified for modern high-performance computers. Since we are focusing on the performance of a single TCP connection running over a wireless channel, we do not consider here competing traffic effects and assume that data are transmitted only in one direction: from the sender to the receiver (see Fig. 1).

Let the service rate of the wireless bottleneck link be μ bits per second and the buffer size at the intermediate system be B IP packets, where all IP packets are of the same size and consist of *MTU* bits. The queue at the intermediate system is the queue of IP packets waiting to be transmitted over the wireless channel. The queue management is assumed to be FIFO Drop-Tail, so any packet arriving when the buffer is full will be lost.

Data packets are buffered at the IP layer of the intermediate system and transmitted one after another to the data-link layer. Once a current packet is successfully delivered next packet is taken from the buffer at the IP layer. At the data-link layer, IP packets are segmented into equal-sized frames. ARQ at the data-link layer is assumed to be completely reliable meaning that a frame is always delivered irrespective of the number of retransmissions it takes. However, if the number of data-link retransmissions causes the *RTT* to exceed the current value of the TCP retransmission timeout (*RTO*) [5], the TCP retransmission timer will expire, leading to a spurious TCP timeout followed by unnecessary retransmission of the last window of data and congestion control procedures invocation.

We also assume that the wireless channel in the reverse direction is completely reliable. Indeed, feedback acknowledgements are usually small in size and well protected by FEC code. Thus, we assume that packet losses happen only in the direction from the sender to the receiver due to buffer overflow at the intermediate system. Finally, we assume that ACK-packets are delivered instantaneously over the wireless channel. All these assumptions were used in many studies and found to be appropriate for relatively high-speed wireless channels [6], [7]. Note that the described model is also suitable to represent "ideal" Selective Repeat ARQ scheme as in [8], [9].

3 Service Process of the Wireless Channel

3.1 Bit Error Model

In this paper, we represent bit error process using a covariance-stationary two-state Markov modulated process. Using relatively simple algorithm outlined below it allows to capture first- and second-order statistical characteristics in terms of error rate and lag-1 autocorrelation coefficient. Note that the extension to the case of general finite-state Markov chain (FSMC) and its variants is straightforward [10].

We model the bit error process using a two-state Markov modulated process. Let $\{W_E(l), l = 0,1,\ldots\}$, $W_E(l) \in \{0,1\}$, denote the model with the modulating Markov chain $\{S_E(l), l = 0,1,\ldots\}$, $S_E(l) \in \{0,1\}$. The model is completely defined using the set of matrices $D_E(k)$, $k = 0,1$, containing transition probabilities from state i to state j with or without incorrect reception of a channel symbol. To parameterize a covariance stationary binary process, only mean and lag-1 autocorrelation coefficient have to be captured. In our previous work, we have showed that there is a unique switched Bernoulli process (SBP) matching mean and lag-1 autocorrelation of covariance stationary bit error observations [11]. This model is given by

$$\begin{cases} \alpha_E = (1 - K_E(1))E[W_E], \\ \beta_E = (1 - K_E(1))(1 - E[W_E]), \end{cases} \quad \begin{cases} f_{1,E}(1) = 0, \\ f_{2,E}(1) = 1, \end{cases} \tag{1}$$

where $f_{1,E}(1)$ and $f_{2,E}(1)$ are probabilities of error in states 1 and 2, respectively, α_E and β_E are transition probabilities from state 1 to state 2 and from state 2 to state 1, respectively, $K_E(1)$ is the lag-1 autocorrelation of bit error observations, $E[W_E]$ is the mean of bit error observations. Details of the algorithm are outlined in [11].

3.2 Frame Error Model

Assume that the length of frames is constant and equals to m bits. Consider the stochastic process $\{W_N(n), n = 0,1,\ldots\}$, $W_N(n) \in \{0,1,\ldots m\}$, $n = lm$, describing the number of incorrectly received bits in consecutive bit patterns of length m. This process is doubly stochastic, modulated by the underlying Markov chain $\{S_N(n), n = 0,1,\ldots\}$ and can be completely parameterized via parameters of the bit error process $\{W_E(l), l = 0,1,\ldots\}$ as shown below.

To parameterize $\{W_N(n), n = 0,1,\ldots\}$, we have to determine m-step transition probabilities of the modulating Markov chain $\{S_E(l), l = 0,1,\ldots\}$ with exactly k, $k = 0,1,\ldots,m$, incorrectly received bits. Denote the probability of transition from state i to state j for the Markov chain $\{S_N(n), n = 0,1,\ldots\}$ with exactly k, $k = 0,1,\ldots,m$, incorrectly received bits in a bit pattern of length m by $d_{N,ij}(k) = \Pr\{W_N(n) = k, S_N(n) = j \mid S_N(n-1) = i\}$.

$$\begin{aligned} &D_N(0) = D_E^m(0), \\ &D_N(1) = \sum_{k=m-1}^{0} D_E^{m-k-1}(0) D_E(1) D_E^k(0), \\ &D_N(2) = \sum_{k=0}^{m-2} D_E^k(0) D_E(1) \sum_{i=m-k-2}^{0} D_E^{m-i-k-2}(0) D_E(1) D_E^i(0), \\ &\ldots \\ &D_N(m) = D_E^m(1). \end{aligned} \tag{2}$$

Let the set of matrices $D_N(k)$, $k = 0,1,\ldots,m$, contains these transition probabilities. These matrices can be found using $D_E(k)$, $k = 0,1$, as given in (2), were $D_N(i)$, $i = 3,4,\ldots,m-2$, can be obtained by induction from $D_N(1)$ or $D_N(m-1)$. The easiest way is to induce $D_N(i)$, $i = 2,3,\ldots,\lfloor m/2 \rfloor$, from $D_N(1)$ and $D_N(i)$, $i = m-2, m-3,\ldots,\lceil m/2 \rceil$, from $D_N(m-1)$.

Note that computation according to (2) is a challenging task and becomes impossible when m is large. Instead, one may use the recursive method as outlined below.

Let us extend the definition of $D_N(k)$, $k = 0,1,\ldots,m$, as follows. We denote the probability of transition from state i to state j for the Markov chain $\{S_N(n), n = 0,1,\ldots\}$ with exactly k, $k = 0,1,\ldots,m$, incorrectly received bits in a bit pattern of length m by $d_{N,ij}(k,m)$. Let the set of matrices $D_N(k,m)$, $k = 0,1,\ldots,m$, contains these transition probabilities. Since at most two errors may occur in two consecutive slots, we have the following expression for $D_N(i,2)$, $i = 0,1,2$:

$$D_N(i,2) = \sum_{k=0}^{i} D_E(k) D_E(i-k), \quad i = 0,1,2, \tag{3}$$

where $D_E(2)$ is the matrix of zeros. Recursively, we get

$$\begin{aligned} D_N(i,3) &= \sum_{k=0}^{i} D_N(k,2) D_E(i-k), && i = 0,1,\ldots 3, \\ D_N(i,4) &= \sum_{k=0}^{i} D_N(k,3) D_E(i-k), && i = 0,1,\ldots 4, \\ &\ldots \\ D_N(i,m) &= \sum_{k=0}^{i} D_N(k,m-1) D_E(i-k), && i = 0,1,\ldots,m, \end{aligned} \tag{4}$$

where $D_E(k)$, $k \geq 2$, and $D_N(i,m)$, $i \geq m+1$, are all zero matrices. Taking this into account we finally have

$$D_N(i,k) = \begin{cases} D_N(i,k-1) D_N(0,1), & i = 0, \\ D_N(i,k-1) D_N(0,1) + D_N(i-1,k-1) D_N(1,1), & i \neq 0, \end{cases} \tag{5}$$

where $D_N(i,1) = D_E(i)$, $i = 0,1$.

The latter equation gives $D_N(k)$, $k = 0,1,\ldots,m$, for a given m. Using the proposed approach the computational complexity decreases significantly and the model can be used for large values of m.

Consider now the frame error process $\{W_F(n), n = 0,1,\ldots\}$, $W_F(n) \in \{0,1\}$, where "0" indicates the correct reception of a frame, "1" denotes the incorrect frame reception. Let us denote the transition probability from state i to state j for the Markov chain $\{S_F(n), n = 0,1,\ldots\}$ with exactly k, $k = 0,1$, incorrectly received frames by $d_{F,ij}(k)$, $k = 0,1$. These probabilities are then combined in the matrices $D_F(0)$ and $D_F(1)$. The process $\{W_N(n), n = 0,1,\ldots\}$, $W_N(n) \in \{0,1,\ldots,m\}$, describing the number of bit errors in consecutive frames is related to the frame error process $\{W_F(n), n = 0,1,\ldots\}$, $W_F(n) \in \{0,1,\ldots,m\}$, as follows:

$$D_F(0) = \sum_{k=0}^{F_T-1} D_N(k), \quad D_F(0) = \sum_{k=F_T}^{m} D_N(k), \tag{6}$$

where F_T is the so-called frame error threshold determining whether a certain frame is correctly received or not. Expressions (6) are interpreted as follows: if the number of incorrectly received bits in a frame is greater or equal to a computed value of the frame error threshold ($k \geq F_T$), then the frame is incorrectly received and $W_F(n) = 1$. Otherwise ($k < F_T$), it is correctly received and $W_F(n) = 0$.

Assume now that the number of bit errors that can be corrected by a FEC code in a frame of length m is l. Then the frame error threshold is $F_T = l + 1$ and the frame is incorrectly received whenever $k \geq F_T$. Otherwise, it is correctly received. Thus, the transition probability matrices (6) take the following form:

$$D_F(0) = \sum_{k=0}^{F_T-1} D_N(k), \quad D_F(1) = \sum_{k=F_T}^{m} D_N(k). \tag{7}$$

3.3 Packet Service Process

Consider now the service process of IP packets at the data-link layer. We assume that each frame requires exactly a unit time to be transmitted over the wireless channel. All IP packets are of the same length and segmented to v frames at the data-link layer. This assumption is not restrictive as data-link frames are usually of fixed size. Due to impairments introduced by wireless transmission medium successful delivery of frames may take random duration in time. As a result, successful delivery of an IP packet is also a random variable with a certain distribution. To parameterize the service process of an IP packet we have to find its service time distribution.

Let $f_P(k)$, $k = m, m+1, \ldots$, be the probability function (PF) of the delay of an IP packet. Consider the IP packet service process describing the number of slots required to successfully transmit a single IP packet. In order to correctly transmit an IP packet, we have to correctly transmit all the frames to which this packet is segmented. Thus, the minimum time to transmit an IP packet is the time to successfully transmit all m frames from first attempts. Since the data-link layer is completely reliable the maximum transmission time of the packet is virtually unlimited.

Firstly, consider the situation when all the frames in a packet are correctly transmitted in their first attempts. In this case, the duration of packet transmission is exactly v slots. In order for service time of the packet to be $(v+1)$ slots, there should be exactly one incorrectly transmitted frame in a sequence of v frames. Generalizing to $(v+i)$ slots delay, we note that there should be exactly i transmission attempts that failed to transmit a frame correctly. Since any packet transmission should end with correctly received frame, the last case can be interpreted as having $(v+i-1)$ errors in $(v+i)$ transmission attempts. The sequence of frames transmissions in a packet is illustrated in Fig. 2, where grey rectangles denote incorrect frame reception and white rectangles stand for correct frame reception.

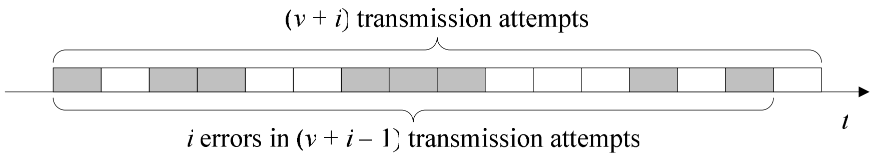

(v + i) transmission attempts
i errors in (v + i – 1) transmission attempts
t

Fig. 2. Sequence of frames transmissions in a single IP packet

Let $\{W_{i,P}(u), u=0,1,\ldots\}$, $i=v,v+1,\ldots$, $W_{i,N}(u)\in\{0,1,\ldots i\}$, $u=ni$, be the set of stochastic processes describing the number of incorrectly received frames in the sequence of i, $i=v,v+1,\ldots$, attempts. Each of these processes contributes one term to the PF describing the transmission time of an IP packet over the wireless channel. For example, $W_{v+i,N}(u)=i$ describes the case when the delay of a packet consisting of v frames is exactly $(v+i)$ slots while there are i incorrectly received frames during the transmission. As a result, in order to determine $f_P(k)$, $k=m,m+1,\ldots$, we have to find out probabilities $W_{v+i,N}(u)=i$ for all $\{W_{i,P}(u), u=0,1,\ldots\}$, $i=v,v+1,\ldots$. To do so, we have to determine probabilistic characteristics of these processes. These processes are doubly stochastic, modulated by the underlying Markov chain $\{S_{i,P}(u), u=0,1,\ldots\}$, and can be completely parameterized via parameters of the frame error processes $\{W_F(n), n=0,1,\ldots\}$.

To parameterize $\{W_{i,P}(u), u=0,1,\ldots\}$, $i=v,v+1,\ldots$, for each process, we have to determine i-step transition probabilities between of the modulating Markov chain $\{S_F(n), n=0,1,\ldots\}$ with exactly k, $k=0,1,\ldots,m$, incorrectly received bits. Let the set of matrices $D_{i,P}(k)$, $k=0,1,\ldots,i$, contains transition probabilities from state i to state j for the Markov chain $\{S_{i,P}(u), u=0,1,\ldots\}$ with exactly k, $k=0,1,\ldots,i$, incorrectly received frames in a frame pattern of length i. These matrices can be found using $D_F(k)$, $k=0,1,\ldots$, similarly to (2).

Note that computation of $D_{i,P}(k)$, $k=0,1,\ldots,i$, is a challenging task that becomes impossible when i is large. Instead, we propose to use the recursive algorithm explained below. Let $d_{P,ij}(k,m)$ be the probability of transition from state i to state j for the Markov chain $\{S_F(n), n=0,1,\ldots\}$ with exactly k, $k=0,1,\ldots,m$, incorrectly received frames in a frame pattern of length m, $m=v,v+1,\ldots$. Let the set of matrices $D_P(k,m)$ contains these transition probabilities. Setting $D_P(k,1)=D_F(k)$, $k=0,1$, we find $D_P(k,m)$ using $D_F(k)$, $k=0,1$, as

$$D_P(i,k)=\begin{cases} D_P(i,k-1)D_P(0,1), & i=0, \\ D_P(i,k-1)D_P(0,1)+D_P(i-1,k-1)D_P(1,1), & i\neq 0. \end{cases} \tag{8}$$

Recalling that the successful packet transmission is only possible when the last frame is also successfully transmitted, the expression for $f_P(k)$, $k=m,m+1,\ldots$, is

$$f_P(k) = \vec{\pi}_k \left(D_P(k-v, k-1) D_F(0) \right) \vec{e}, \tag{9}$$

where $\vec{e}$ is the vector of ones of appropriate size, $\vec{\pi}_k$ is the steady state probability vector of $\{W_{k,P}(n), n = 0,1,\dots\}$. Vectors $\vec{\pi}_k$, $k = v, v+1, \dots$, can be found as the solution of the following matrix equations: $\vec{\pi}_k D_F^k = \vec{\pi}_k$, $\vec{\pi}_k \vec{e} = 1$.

4 TCP SACK Model

In this section, we consider the evolution of a TCP SACK connection and derive expression for its long-term steady state goodput, where the goodput is the useful number of bits received by the destination per second. The developed model is based on the fluid model approach, first proposed in [12].

Let the round-trip path delay of the wired network be τ seconds. The system performance bottleneck is determined by the limited service rate of the wireless channel. In contrast to the wired network, where multiple packets can be sent without waiting for the first packet to reach the other end of the link, the wireless channel cannot hold multiple packets in the air at once since the previously transmitted packet should be successfully delivered before starting a new transmission. Thus, the bandwidth-delay product of the network path can be found as $C = 1 + \mu\tau / MTU$. The maximum number of IP packets that can be accommodated in the network is $(C + B)$, assuming that there are C packets in flight and the buffer at the intermediate system is fully occupied. Since $B \geq 1$, it implies that $C + B \geq 3$.

Let us consider steady state TCP SACK behavior in the absence of delay spikes caused by wireless channel impairments. In this case, the connection experiences periodic packet losses: each time *cwnd* exceeds the maximum number of packets that can be accommodated in the network, the last packet in a window of data is dropped due to buffer overflow at the intermediate system. According to the sliding window algorithm, after the lost segment $(C + B)$ more segments are sent, triggering duplicate ACKs. As long as $C + B \geq 3$, the sender receives enough duplicate ACKs to trigger the loss recovery algorithm [4]. Thus, the cyclical evolution of TCP SACK is as follows. A cycle starts after a segment loss is detected via three duplicate ACKs. Then the current *cwnd* is set to roughly $(C + B)/2$ and the congestion avoidance phase begins. The receiver sends one ACK for every b-th segment it gets, so *cwnd* increases linearly with a slope of $1/b$ segments per *RTT* until $cwnd = C + B$ (see Fig. 3). Factor b depends on the acknowledgement strategy of the receiver: as specified in [13], a TCP should use delayed ACKs, sending an ACK for at least every second data segment ($b = 2$); however, it is also allowed to send an ACK for every data segment ($b = 1$). The next additive increase of *cwnd* leads to new buffer overflow and the current cycle ends. Thus, we consider a cycle to be a period between two consecutive losses. Since any lost segment can be recovered within a single *RTT* by using the SACK-based loss recovery algorithm, we are neglecting the details of the loss recovery phase between cycles as having negligible effect on TCP SACK performance.

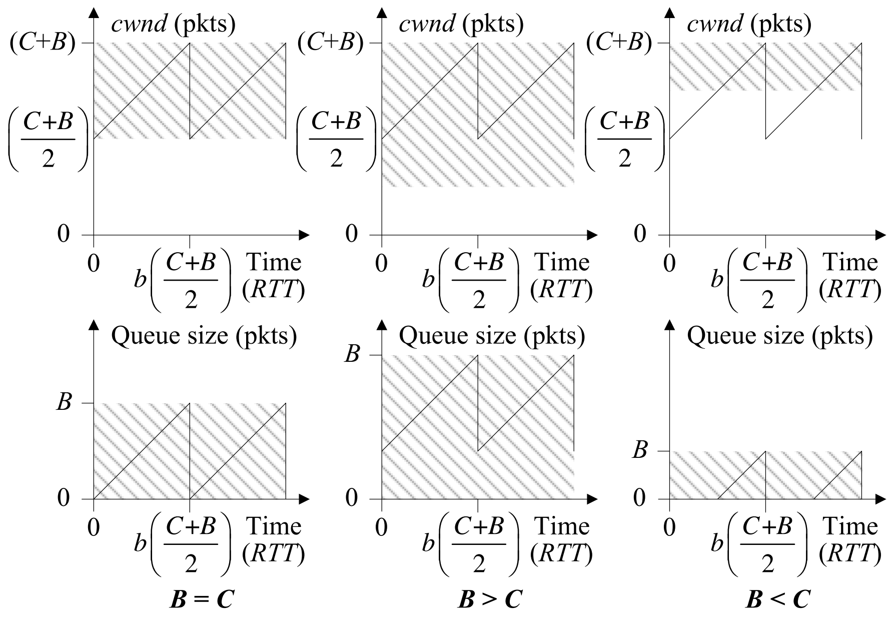


Fig. 3. TCP SACK window evolution and buffer occupancy under periodic packet losses

The duration of a cycle and the number of segments successfully delivered during a cycle can be defined as

$$A = \overline{RTT} \cdot b\left(\frac{C+B}{2}\right), \quad Y = b\left(\frac{C+B}{2}\right)^2 + \frac{b}{2}\left(\frac{C+B}{2}\right)^2 = \frac{3b}{2}\left(\frac{C+B}{2}\right)^2, \tag{10}$$

Note that the *RTT* is given by two components: the round-trip path delay τ of the wired network and the queuing delay. While τ is a static measure of the physical distance from the sender to the intermediate system and back, the queuing delay depends on the current value of *cwnd*, the time required to transmit a single IP packet over the wireless channel, and the ratio between the buffer size B and the bandwidth-delay product C of the network path (see Fig. 3). Note that the buffer occupancy roughly follows the saw-tooth TCP SACK window evolution. Thus, the average queuing delay δ can be found by multiplying the average queue size R during a cycle by the average time ε required to transmit a single IP packet over the wireless channel:

$$\overline{RTT} = \tau + \delta, \quad \varepsilon = \frac{m}{\mu}\sum_{k=v}^{\infty} f_P(k)k, \quad \delta = R\varepsilon = \begin{cases} \left(\dfrac{3B-C}{4}\right)\varepsilon, & B \geq C, \\ \left(\dfrac{B^2}{C+B}\right)\varepsilon, & B < C. \end{cases} \tag{11}$$

Let us consider the effect of delay spikes on the long-term steady state goodput of a TCP SACK connection. As it was pointed out in [1], a TCP spurious timeout occurs when the *RTT* value suddenly increases to the extent that it exceeds the duration of the TCP retransmission timer, *RTO*. In the considered scenario, the completely reliable data-link layer can cause a sudden delay due to bit errors on the wireless channel.

Similarly to [14], we consider the evolution of a TCP SACK connection as a sequence of supercycles, where a supercycle is a period between two consecutive spurious timeouts (see Fig. 4). When the variability in wireless channel quality introduces a sudden delay in the service process of an IP packet, all the subsequent transmissions up to the end of the delay spike will be delayed as well. After the TCP retransmission timer expiration, the sender retransmits the first unacknowledged segment and in the absence of any feedback from the receiver it will continue trying to deliver this segment as specified in [5]. Shortly after the idle period, the ACK for the original transmission returns to the sender. On receipt of this ACK after the wireless channel outage, the TCP SACK sender mistakenly interprets this ACK as acknowledging the previously retransmitted segment and enters the slow start phase with unnecessary retransmission of all other outstanding segments in the Go-Back-N method [15].

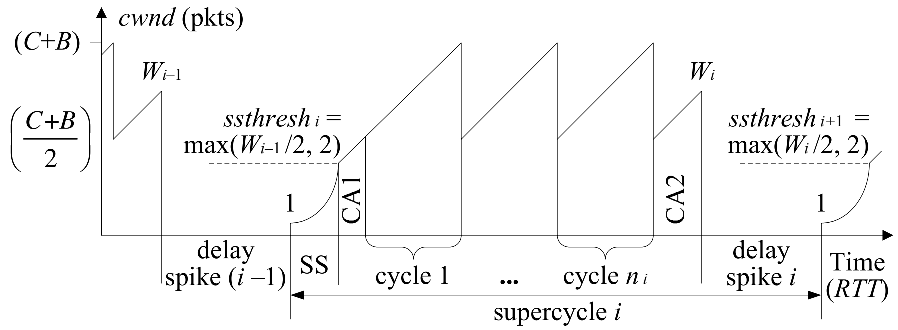


Fig. 4. TCP SACK window evolution in the presence of delay spikes

Let W_{i-1} denote the window size when delay spike $(i-1)$ occurs. After the TCP retransmission timer expiration, the slow start threshold and the current value of *cwnd* will be set as $ssthresh_i = \max(W_{i-1}/2, 2)$ and $cwnd = 1$. Assuming that delay spikes are less frequent than packet losses due to buffer overflow, we can safely assume that random variable W_i is uniformly distributed from $(C+B)/2$ to $(C+B)$. Hence

$$E[W] = 3(C+B)/4, \quad E[ssthresh] = \max(E[W]/2, 2). \tag{12}$$

The expected durations of phases CA1 and CA2 can be found as

$$\begin{aligned} E\left[A^{CA1}\right] &= \overline{RTT} \cdot b\left(\frac{C+B}{2} - \frac{E[W]}{2}\right) = \overline{RTT} \cdot b\left(\frac{C+B}{8}\right), \\ E\left[A^{CA2}\right] &= \overline{RTT} \cdot b\left(E[W] - \frac{C+B}{2}\right) = \overline{RTT} \cdot b\left(\frac{C+B}{4}\right), \end{aligned} \tag{13}$$

and the expected number of segments delivered during these phases can be defined as

$$E\left[Y^{CA1}\right]=b\left(\frac{C+B}{2}-\frac{E[W]}{2}\right)\frac{E[W]}{2}+\frac{b}{2}\left(\frac{C+B}{2}-\frac{E[W]}{2}\right)^2=\frac{7b}{2}\left(\frac{C+B}{8}\right)^2,$$
$$E\left[Y^{CA2}\right]=b\left(E[W]-\frac{C+B}{2}\right)\left(\frac{C+B}{2}\right)+\frac{b}{2}\left(E[W]-\frac{C+B}{2}\right)^2=\frac{5b}{2}\left(\frac{C+B}{4}\right)^2. \quad (14)$$

The number of segments delivered during the slow start (SS) phase can be closely approximated as a geometric series $Y_i^{SS}=1+\gamma+\gamma^2+\ldots+\gamma^{N_i-1}=(\gamma^{N_i}-1)/(\gamma-1)$, where $\gamma=1+1/b$ [16]. Taking into account that in the slow start phase of the i-th supercycle *cwnd* growths exponentially from 1 to $ssthresh_i$, we get that $\gamma^{N_i-1}=\max\left(W_{i-1}/2,2\right)$. Hence the expected duration of the slow start phase and the number of segments successfully delivered during this phase can be expressed as

$$E\left[A^{SS}\right]=\overline{RTT}\cdot\max\left(\log_\gamma\left(\frac{\gamma E[W]}{2}\right),2\right),\quad E\left[Y^{SS}\right]=\max\left(\frac{\gamma E[W]-2}{2(\gamma-1)},3\right). \quad (15)$$

Combining (10)-(15), we define TCP SACK long-term steady state goodput as

$$G=\frac{MSS\left(Y+Q\left(E[Y^{SS}]+E[Y^{CA1}]+E[Y^{CA2}]-(E[W]-1)\right)\right)}{A+Q\left(E[T]+E[A^{SS}]+E[A^{CA1}]+E[A^{CA2}]\right)}, \quad (16)$$

where Q is the probability of TCP retransmission timer expiration, $(E[W]-1)$ is the expected number of spuriously retransmitted segments during the slow start phase, $E[T]$ is the expected duration of a delay spike. Using (9) we have

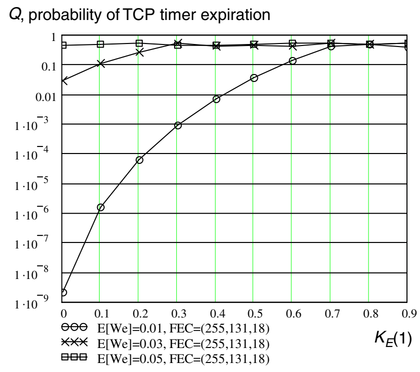
$$E[T]=\frac{m}{\mu}\sum_{k=n\lceil\varepsilon\rceil}^{\infty}f_P(k)k,\quad Q=\sum_{k=n\lceil\varepsilon\rceil}^{\infty}f_P(k), \quad (17)$$

where n, $n\geq 1$, relates to the granularity of the TCP retransmission timer.

5 Numerical Analysis

Firstly, let us consider whether delay spikes often occur on the wireless channels behaving in stationary manner. In order to do so, we estimate probabilities of delay spike for different input parameters. In what follows, to visualize the effect of different parameters we choose the following settings: $E[W_E]\in\{0.01,0.02,\ldots,0.09\}$, $K_E(1)\in\{0.0,0.1,\ldots,0.9\}$, (255,131,18) and (255,87,26) BCH FEC codes, $MTU=1500$ bytes, $MSS=1460$ bytes, $\mu=384$ kbit/s, $\tau=10$ ms, $B=80$ packets.

Probabilities of TCP retransmission timer expiration for $n=1$ (see (17)) are shown in Fig. 5. First of all, we note that these dependencies are not monotonic. The reason is that the probability function (PF) of the delay of a packet is discrete in nature. Thus, depending of the value of the mean delay the probability of TCP retransmission timer expiration computed according to (17) may slightly vary. Indeed, this probability is just mass of the PF for all k which are greater than $\lceil E[f_P] \rceil$. When $E[f_P]$ is closer to $\lfloor E[f_P] \rfloor$ than to $\lceil E[f_P] \rceil$, there is more probability mass in the right hand part of the distribution. Next, we note that the maximum probability of TCP retransmission timer expiration for $n=1$ may approach 0.5. This is explained by the structure of PFs of the delay of a packet shown in Fig. 6. As one may notice, when the bit error rate increases, PF tends to distribution which is symmetric around its mean. Since for $n=1$ the probability of TCP retransmission timer expiration is the probability that delay spike is greater than $\lceil E[f_P] \rceil$, depending on the skewness of PF, it may approach or even be greater than 0.5. Concerning general dependencies we note that the probability of TCP retransmission timer expiration increases as bit error rate and lag-1 autocorrelation increase. The effect of FEC is also noticeable. For (255,87,26) FEC code probability of delay spike is less compared to (255,131,18) FEC code.



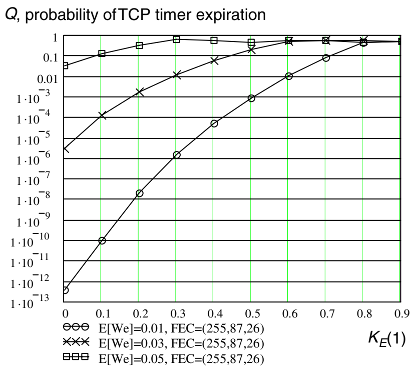
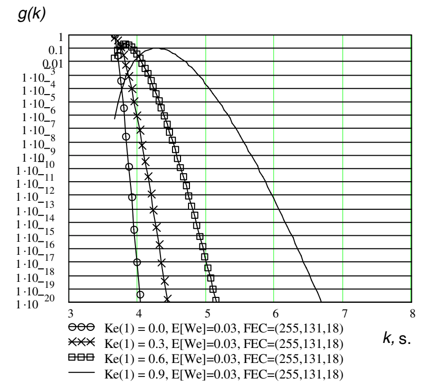


Fig. 5. Probabilities of TCP retransmission timer expiration for $n=1$



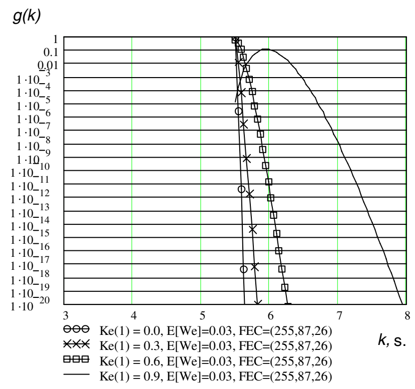
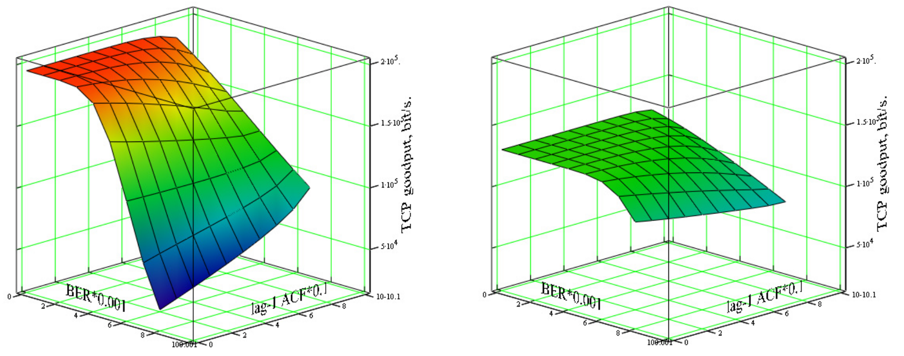


Fig. 6. Probability functions of IP packet delay for different $K_E(1)$

It is important to note that probability of TCP retransmission timer expiration is negligible when n is greater than 1. For example, already for $n = 1.5$ the probability of TCP retransmission timer expiration is less than 10E-15 for $E[W_E] = 0.09$, $K_E(1) = 0.9$, and both FEC codes. In accordance with [5], n should be always greater than 1. Moreover, coarse-grained (500 ms) clocks, traditionally used by TCP implementations to measure the *RTT* and compute the *RTO*, also lead to large values of n (e.g., $n \geq 2$).

TCP SACK long-term steady state goodput as a function of $E[W_E]$, $K_E(1)$, and FEC code is shown in Fig. 7. Let us fix the FEC code to (255,131,18) and consider what happens when $E[W_E]$ and $K_E(1)$ vary. When bit error rate is less than 0.6, the increase in lag-1 ACF values leads to worse performance of the wireless channel. Indeed, higher correlation leads to more lengthy bursts of errors within a single frame that FEC code cannot conceal. This, in turn, increases the number of retransmission attempts required to successfully transmit an IP packet over the wireless channel affecting the *RTT*. However, when error rate becomes higher, high values of lag-1 ACF result in better performance. This effect is due to limitations of a given FEC. Indeed, with these rates, bit errors, even well distributed in time, result in more incorrectly received channel symbols per frame than this FEC code can handle. This increases the *RTT* and, therefore, decreases TCP goodput for weak channel correlation. On the other hand, due to error grouping effect, higher values of lag-1 ACF lead to better TCP goodput. The situation changes when the strength of FEC code increases. Note that for small values of bit error rate (up to 0.05) TCP goodput for (255,87,26) FEC code is less than TCP goodput for (255,131,18) FEC code. However, when error rate increases, the performance gain over (255,131,18) FEC code becomes clear. The effect of correlation remains the same. The only reason why it is invisible is that we do not show here the region of bit error rate values for which higher correlation leads to better performance. However, it is clear that for considered error rates higher lag-1 ACF results in lower TCP goodput. This is the effect of error grouping within a single frame resulting in more incorrectly received frames and thus increasing the *RTT*.



a) FEC=(255,131,18) b) FEC=(255,87,26)

Fig. 7. TCP SACK steady state goodput as a function of $E[W_E]$, $K_E(1)$, and FEC code

6 Conclusion

In this paper, we proposed an analytical model for a TCP SACK connection running over a covariance stationary wireless channel with completely reliable ARQ/FEC. The proposed model allows to evaluate the effect of many parameters of wireless channels on TCP performance making it suitable for performance optimization studies. These parameters include stochastic properties of wireless channel characteristics, the size of protocol data units at different layers, the strength of FEC code, the presence of ARQ, the buffer size at the IP layer, and the service rate of the wireless channel. Among other conclusions, we show that TCP spurious timeouts do not occur when wireless channel conditions are stationary.

The developed model is a general framework rather than a model for particular wireless access technology. To use it in practical evaluation of different technologies this framework should be extended by adding particular details of state-of-the-art wireless systems.

References

1. Gurtov, A.: Efficient Data Transport in Wireless Overlay Networks. Ph.D. Thesis, University of Helsinki, Finland (2004)
2. Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: TCP Selective Acknowledgement Options. RFC 2018 (1996)
3. Floyd, S., Mahdavi, J., Mathis, M., Podolsky, M.: An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC 2883 (2000)
4. Blanton, E., Allman, M., Fall, K., Wang, L.: A Conservative Selective Acknowledgement (SACK)-based Loss Recovery Algorithm for TCP. RFC 3517 (2003)
5. Paxson, V., Allman, M.: Computing TCP's Retransmission Timer. RFC 2988 (2000)
6. Zorzi, M., Rao, R., Milstein, L.: ARQ Error Control for Fading Mobile Radio Channels. IEEE Transactions on Vehicular Technology 46(2), 445–455 (1997)
7. Zorzi, M., Rao, R.: Throughput Analysis of Go-Back-N ARQ in Markov Channels with Unreliable Feedback. IEEE ICC, 1232–1237 (1995)
8. Krunz, M., Kim, J.-G.: Fluid Analysis of Delay and Packet Discard Performance for QoS Support in Wireless Networks. IEEE JSAC 19(2), 384–395 (2001)
9. Fantacci, A.: Queuing Analysis of the Selective Repeat Automatic Repeat Request Protocol for Wireless Packet Networks. IEEE Transactions on Vehicular Technology 45(2), 258–264 (1996)
10. Swarts, J., Ferreira, H.: On the Evaluation and Application of Markov Channel Models in Wireless Communications. IEEE VTC 1, 117–121 (1999)
11. Moltchanov, D., Koucheryavy, Y., Harju, J.: Simple, Accurate and Computationally Efficient Wireless Channel Modeling Algorithm. In: Braun, T., Carle, G., Koucheryavy, Y., Tsaoussidis, V. (eds.) WWIC 2005. LNCS, vol. 3510, pp. 234–245. Springer, Heidelberg (2005)
12. Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. Computer Communication Review 27(3), 67–82 (1997)
13. Braden, R. (ed.): Requirements for Internet Hosts. RFC 1122 (1989)
14. Fu, S., Atiquzzaman, M.: Modelling TCP Reno with Spurious Timeouts in Wireless Mobile Environments. ICCCN, 391–396 (2003)
15. Guan, Y., et al.: Simulation Study of TCP Eifel Algorithms. OPNETWORK (2005)
16. Cardwell, N., Savage, S., Anderson, T.: Modeling TCP Latency. In: IEEE INFOCOM, pp. 1742–1751 (2000)

Flow Level Performance Comparison of Packet Scheduling Schemes for UMTS EUL

D.C. Dimitrova[1,⋆], Hans van den Berg[1,2], G. Heijenk[1], and R. Litjens[2]

[1] University of Twente, Enschede, The Netherlands
d.c.dimitrova@ewi.utwente.nl
[2] TNO ICT, Delft, The Netherlands

Abstract. The Enhanced Uplink (EUL) is expected to provide higher capacity, increased data rates and smaller latency on the communication link from users towards the network. A key mechanism in the EUL traffic handling is the packet scheduler. In this paper we present a performance comparison of three distinct EUL scheduling schemes (one-by-one, partial parallel and full parallel) taking into account both the packet level characteristics and the flow level dynamics due to the random user behavior. For that purpose, we develop a hybrid analytical/simulation approach allowing for fast evaluation of performance measures such as mean file transfer time and fairness expressing how the performance depends on the user's location.

1 Introduction

With the specification of the Enhanced Uplink (EUL) in 3GPP Release 6 of the UMTS standard [2] a next step in the evolution of WCDMA-based cellular networks is made. As the uplink counterpart of the HSDPA (High Speed Downlink Packet Access) technology standardised in 3GPP Release 5 [1] and currently being introduced by many mobile operators, EUL is primarily designed for better support of elastic data applications.

The enhanced uplink introduces a new transport channel called EDCH, see e.g. [7]. Channel access is coordinated by the base stations via packet scheduling based on time frames of fixed length (2 or 10 ms, termed TTI: Transmission Time Interval). Fast rate adaptation with an enhanced dynamic range and efficient time multiplexing through appropriate scheduling schemes enable higher data transfer rates than usually provided on DCHs in 'plain' UMTS. Other key benefits offered by the EUL technology are an enhanced cell capacity and a reduced latency. In contrast to HSDPA for the downlink, due to limited transmit powers of the user terminals, a single uplink user cannot always use the total available channel resource on its own when it is scheduled (which would optimize throughput, cf. [12]) depending on its distance to the base station. Hence, it makes sense to consider scheduling schemes with simultaneous transmissions on the uplink, see e.g. [7].

⋆ Corresponding author.

In the present research we compare the performance of different EUL scheduling schemes, as we are particularly interested in the influence of flow level dynamics due to flow (file) transfer completions and initiations by the users at random time instants, which leads to time varying number of ongoing flow transfers. We aim at quantifying performance measures such as file transfer times and fairness, expressing how the performance depends on the user's location in the cell.

Most EUL performance studies in literature are based on dynamic system simulations, see e.g. [13], [5], [10]. The underlying simulation models incorporate many details of the channel operations and traffic behaviour, but running the simulations tends to require a lot of time. Analytical modelling may overcome this problem by abstracting from system details yet allowing the same qualitative insights into the system performance. Most analytical studies focus on the performance of schedulers without taking into account the impact of the flow level dynamics, see e.g. [9]. Analytical studies on EUL performance capturing both the packet and flow level dynamics of the system are rare. Interesting references here are [4] and [11]. In particular, in [11] flow level performance metrics are analysed for two (rate-fair) scheduling disciplines assuming that the transmit powers of all mobiles are sufficient to reach the maximum bit rate.

In the present paper we extend the model in [11] to the practical situation that the transmit power of the users is a limiting factor and scheduling schemes are not rate-fair per se but access-fair which may, implicitly, favour users close to the base station over users at the cell edge. We consider a single cell scenario with two types of users: EUL (EDCH) users generating elastic traffic flows and DCH users generating traffic flows (e.g. speech calls) that require a constant bit rate. EUL traffic is of best effort type and adapts to the available resources left over by the interfering DCH traffic.

Our modelling and analysis approach is based on time scale decomposition and consists basically of three steps. The first two steps take the details of the scheduler's behaviour into account in a given state of the system, i.e. the number of EDCH and DCH users and their distance to the base station. In particular, in the first step the data rate at which a scheduled EDCH user can transmit is determined. The second step determines the user's average throughput by accounting for the frequency at which the user is scheduled for transmitting data. In the third step these throughputs and the rates at which new DCH and EDCH users become active are used to create a continuous-time Markov chain describing the system behaviour at flow level. From the steady-state distribution of the Markov chain the performance measures, such as mean file transfer time of a user, can be calculated.

Due to the complexity of the resulting Markov model (transition rates are dependent on the full state) an analytical solution is not feasible; only for some special cases explicit expressions can be obtained for the steady-state distribution. When such closed-form expressions are not available, standard techniques for deriving the steady-state distribution can be used, e.g. numerical solution of the balance equations or simulation of the Markov chain. As the jumps in the Markov chain only apply to the initiation or completion of flow transfers (note that the packet level details are captured in the transition rates which are calculated analytically), simulation of the Markov chain is a very attractive option and does not suffer from the long running times of the detailed system simulations used in many other studies.

The rest of the paper is organized as follows. Section 2 introduces the three different scheduling schemes we will analyse in this paper. In Section 3 we describe the network scenario considered in this paper and state the modeling assumptions. Subsequently, in Section 4 the anaytical performance evaluation approach is described in general terms, while the details of the analysis for each of the three scheduling schemes are given in Section 5. Section 6 presents and discusses numerical results illustrating their performance. Finally, in Section 7, conclusions and our plans for future work are given.

2 Scheduling Schemes for Enhanced Uplink

In this paper we focus on a class of scheduling schemes for the enhanced uplink, where the users get fair channel access independent of the actual channel conditions (channel 'oblivious' scheduling). Three different schedulers are investigated, termed one-by-one (OBO), partial parallel (PP) and full parallel (FP), which will be described in more detail below. The strategies mainly differ in the time scale on which the fair access is effectuated, in particular whether this is done within each TTI separately, or over a so-called scheduling cycle of multiple TTIs. Note, that fair channel access does not necessarily imply that each scheduling scheme yields equal bit rates to the different active EDCH users. The experienced bit rates depend on the received powers which can be different due to different distances to the serving base station.

A common notion in the three schemes is the available channel resource, termed total received power budget (B) at the base station. Expressed in linear units, B is the product of the noise rise target at the base station and the thermal noise. Part of the total budget B cannot be used by the EDCH users in a cell because of interference generated by other sources, e.g. thermal noise, intra-cell interference generated by DCH users and inter-cell interference generated by (E)DCH users in other cells. The budget left over for the intra-cell EDCH users (which varies over time) is termed the EDCH budget denoted by B'. We will now describe the scheduling schemes considered in this paper in more detail.

One-by-one (OBO) Scheduler

In this scheme, during a TTI, a single EDCH user is allowed to transmit and may use the entire available EDCH budget. The different EDCH users are selected for transmission in subsequent TTIs in a round robin fashion [11]. Figure 1(a) illustrates this scheme. Although it is generally beneficial to schedule only a single transmission during a TTI, due to high achievable instantaneous rate [12], there is also a downside, as a single EDCH user may not be able to fully utilize the available EDCH budget because of its power limitations. What part of the available resources is unused depends on the user's channel conditions. A user

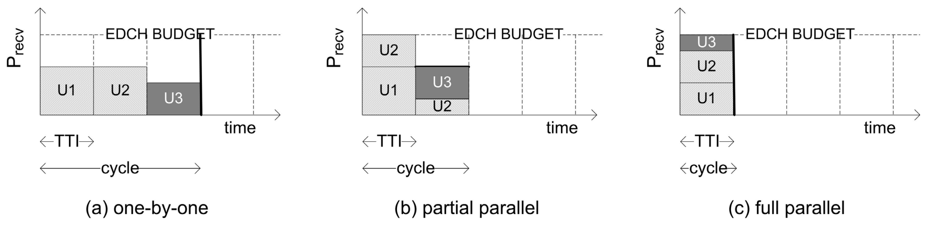


Fig. 1. Illustration of the packet handling of the considered EUL scheduling schemes

with good channel conditions, typically located close to the base station, is able to generate a relatively high received power level and hence leave less resources unutilised.

Partial Parallel (PP) Scheduler
The PP strategy attempts to optimize the strategy underlying the OBO scheme by selecting additional EDCH users for simultaneous transmission when the available EDCH budget cannot be fully utilised by a single transmission, see Figure 1(b). In fact, for a given TTI, EDCH users are added for simultaneous transmission as long as the sum of their maximum received powers does not exceed the available EDCH budget; the remainder of the budget is filled up by an additional EDCH user whose transmission is split over two consecutive TTIs[1]. Overall, the user selection in consecutive TTIs is done in a round robin fashion yielding fair channel access for the users.

Full Parallel (FP) Scheduler
The last scheduling scheme considered in this paper is the FP scheme (see e.g. [11]) which, like the PP scheme, also aims at full utilization of the channel resource. In this scheme all EDCH users are given simultaneous channel access in a given TTI, see Figure 1(c). If the total amount of resources requested by the EDCH users (when transmitting at their maximum power) is larger than the available EDCH budget, then the transmit powers are decreased proportionally.

In a preliminary qualitative performance comparison of the three scheduling schemes we expect the PP scheduler to perform best. Which of the other two schedulers is best primarily depends on the available budget: if the budget is relatively low, the OBO scheduler is expected to outperform the FP scheduler since it experiences no interference from other EUL users; if the budget is high, the FP scheduler is likely to better utilize the available budget. In terms of operational complexity and computation the three schedulers differ - OBO being the least complex and PP the most. However, compared to other EUL functionality, e.g. power control, the level of complexity is relatively low.

[1] Obviously, other possible strategies exist to deal with filling up the last part of the available budget, but the differences between various strategies appear to be very small.

3 Modelling Assumptions

In this section we describe the modelling assumptions underlying the presented analysis. At the *system* level, we consider the uplink of a single cell with an omnidirectional base station, serving both DCH and EDCH calls. As illustrated in Figure 2(a), the considered cell is split in K concentric zones, where zone i is characterized by a distance d_i to the base station and a corresponding path loss denoted $L(d_i)$, $i = 1, \cdots, K$. Derived from an operator-specified noise rise target, the total received power budget at the base station is denoted B. This budget is partially consumed by the constant thermal noise level N, while the remainder is consumed by a varying amount of intra-cell interference originating from either DCH or EDCH calls. The DCH budget is the maximum part of the total budget that may be used by DCH calls. At any time, the EDCH calls may fully use that part of the budget that is not claimed by the thermal noise or on-going DCH calls: this is referred to as the EDCH budget, which is denoted $B'(n_D)$, where n_D denotes the number of existing DCH calls.

A number of additional assumptions are made at the *user* level. Calls are generated according to spatially uniform Poisson arrival processes with rates λ (EDCH calls) and λ_D (DCH calls). For the performance of EDCH calls it matters in which zone they appear. As a direct consequence of the uniformity assumption, the probability q_i that a generated EDCH call appears in zone i, is calculated as the ratio of the area of zone i and the total cell area, so that the EDCH call arrival rate in zone i is equal to λq_i, $i = 1, \cdots, K$. EDCH calls are characterised by a file that needs to be uploaded, whose size is exponentially distributed with mean F (in kbits). All calls have the same maximum transmit power $P_{\max}^{tx}$ but different maximum received power at the base station $P_{i,\max}^{rx}$ due to the zone-dependent path loss. As no user mobility is considered, users keep their positions in the cell during the file transmission. The bit rate at which an EDCH call is served depends on the experienced signal-to-interference ratio C/I. Given a prefixed E_b/N_0 (energy-per-bit to interference-plus-noise-density ratio) requirement, the attainable bit rate is equal to $r = r_{chip}\left(C/I\right)/\left(E_b/N_0\right)$, where $r_{chip} = 3840$ kchips/s denotes the system chip rate. The signal level C is determined by the call's transmit power and the zone-dependent path loss. The interference level I comprises several distinct components: *(i)* the thermal noise level N; *(ii)* the self-interference modelled by parameter ω, which is due to the effects of multipath fading; *(iii)* the interference $I_{EDCH}(\underline{n})$ originating from EDCH calls; and *(iv)* the interference $I_{DCH}(n_D)$ originating from DCH calls. DCH calls model e.g. speech telephony or video streaming calls and are characterised by a constant bit rate and hence a prefixed consumption P_D of the base station's received power budget, regardless of the specific location of the user. The applied value of P_D is based on a worst-case assumption that the noise rise target is fully utilised, which is indeed the objective of the EUL scheduler and hence a rather harmless assumption. Using P_D the above-mentioned DCH budget is readily translated to a maximum m on the number of admissible DCH calls, where m is increasing in the DCH budget and decreasing in the bit rate (which determines P_D). Also considering the single cell focus of our study, note

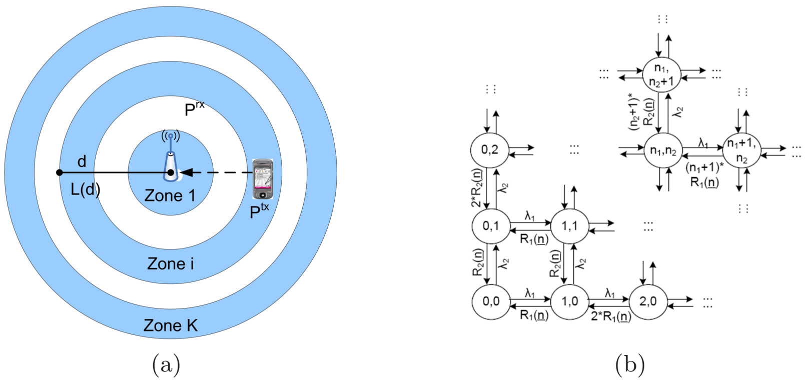


Fig. 2. Modelling approach. Figure (a) shows the split up of the cell into K concentric zones; figure (b) shows the state transition diagram of the continuous-time Markov chain describing the system at flow level for the case $K = 2$ (DCH dimension is excluded for clarity of presentation).

that it suffices to keep track of the *aggregate* number of DCH calls in the cell. The DCH call duration is exponentially distributed with mean τ (in seconds). At a given time, the system state $\underline{n} \equiv (n_1, n_2, \cdots, n_K, n_D)$ is described by the number of EDCH calls n_i in zone i, $i = 1, \cdots, K$, and the total number of DCH calls n_D.

In the considered *tra c handling* scheme the fixed rate DCH calls are treated with priority within the DCH budget, while at any time the EDCH calls are allowed to utilise the remaining part of the uplink budget, including any part of the DCH budget that is not used by DCH calls. Consequently, the dynamics of the DCH calls can be described by an $M/M/m/m$ queueing model (Erlang loss model), which is independent of the EDCH dynamics. For this Erlang loss model explicit expressions are known that relate the traffic load and the channel capacity to the induced blocking probability.

4 Generic Analysis

We now move on to present how the EDCH performance is analysed by applying our proposed three-step approach. The approach here is generic in the sense that it covers all proposed schedulers. In the next section, it will be 'filled in' with the specifics of the different scheduling schemes in order to complete the analysis.

4.1 Instantaneous Rate

In the first step of the analysis we determine the so-called instantaneous bit rate $r_i(\underline{n})$, i.e. the transmission rate a call in zone i can achieve *when* it is scheduled for transmission. The instantaneous rate is defined within the boundaries of a TTI and depends on the zone i where the user is located and the interference experienced from all other calls that are scheduled simultaneously, which in turn depends on the current state $\underline{n}$ and the scheduling scheme. We define $r_i(\underline{n})$ as a generalization of [6] eq. 8.4

$$r_i(\underline{n}) = \frac{r_{chip}}{E_b/N_0} \cdot \frac{C}{I} = \frac{r_{chip}}{E_b/N_0} \cdot \frac{P_i^{rx}}{I_{EDCH}(\underline{n}) - \omega P_i^{rx} + I_{DCH}(n_D) + N}. \qquad (1)$$

Since $I_{EDCH}(\underline{n})$ is defined to include the reference call's own signal, a fraction ω of the own signal must be subtracted from $I_{EDCH}(\underline{n})$ to model the effects of self-interference properly. Since P_i^{rx}, $I_{EDCH}(\underline{n})$ and $I_{DCH}(n_D)$ depend on the state $\underline{n}$ and/or the scheduling scheme, so does the instantaneous rate $r_i(\underline{n})$.

4.2 State-Dependent Throughput

Knowledge of $r_i(\underline{n})$ is not sufficient to determine a call's throughput in system state $\underline{n}$ since a call often has to wait several TTIs between actual data transmissions (scheduling cycle; see also Figure 1). The result is a decreased effective transmission rate, which we term state-dependent throughput $R_i(\underline{n})$. More precisely, $R_i(\underline{n})$ is the average transmission rate an active call achieves during one scheduling cycle, given that the system is and remains in state $\underline{n}$. Denoting with $c(\underline{n})$ the cycle length, which depends on the number of ongoing EDCH calls and the applied scheduling scheme, we have

$$R_i(\underline{n}) = \frac{r_i(\underline{n})}{c(\underline{n})}. \qquad (2)$$

4.3 Markov Chain Modelling

Now that the packet level analysis is completed we can introduce flow level dynamics. This is done in the third step of the analysis with the creation of a continuous-time Markov chain model describing the dynamics of EDCH and DCH call initiations and completions in the cell. The states in the Markov model are given by $\underline{n} = (n_1, n_2, \cdots, n_K, n_D)$, i.e. the distribution of the EDCH calls over the different zones in the cell and the total number of on-going DCH calls. Hence the Markov model itself has $K+1$ dimensions, with K dimensions covering the EDCH calls in the different zones and an additional dimension to cover the DCH calls in the cell. Each of the K 'EDCH dimensions' is unlimited in the number of admissible calls, while the 'DCH dimension' is limited to m simultaneous calls. The transition rates of the Markov model are as follows, cf. Figure 2(b):

$\underline{n} \rightarrow (n_1, \cdots, n_i + 1, \cdots, n_K, n_D)$	at rate	λ_i	(EDCH call arrival)
$\underline{n} \rightarrow (n_1, \cdots, n_{i+1}, \cdots, n_K, n_D + 1)$	at rate	λ_D	(DCH call arrival)
$\underline{n} \rightarrow (n_1, \cdots, n_i - 1, \cdots, n_K, n_D)$	at rate	$\frac{n_i}{F} R_i(\underline{n})$	(EDCH call completion)
$\underline{n} \rightarrow (n_1, \cdots, n_i + 1, \cdots, n_K, n_D - 1)$	at rate	$\frac{n_D}{\tau}$	(DCH call completion)

From the steady-state distribution of the Markov model we can easily derive the desired performance measures, viz. the mean file transfer times for the different zones and the fairness index.

5 Scheduler-Specific Analysis

The generic three-step approach described in Section 4 can be used to analyse various schedulers. Applied to the three scheduling schemes of our interest the approach results in three different Markov models that can be classified as complex processor sharing type of queueing models. The differences are a consequence of the different expressions of $r_i(\underline{n})$ and $c(\underline{n})$. Due to the specifics of the schedulers, the Markov chains generated here are too complex for the steady-state distribution to be obtained analytically. Therefore we have chosen to simulate the Markov model in order to find the steady-state distributions.

5.1 One-by-One (OBO) Scheduler

In case of OBO scheduling only one EDCH call may be scheduled per TTI. Hence the scheduled user may in principle utilise the entire EDCH budget but may very well be limited by its own maximum received power level: $P_i^{rx} = \min\left\{P_{i,\max}^{rx}, B'(n_D)\right\}$. Having only a single active user per TTI yields a cycle length of $n \equiv n_1 + n_2 + \cdots + n_K$ (for state $\underline{n}$), hence, $c(\underline{n}) = n$. Under OBO scheduling the sources of interference are thermal noise, interference from DCH calls and self-interference. $I_{EDCH}(\underline{n})$ in this case consists only of the own signal power, which allows expression (1) to be rewritten and the resulting state-dependent throughput $R_i(\underline{n})$ is given by:

$$R_i(\underline{n}) = \frac{r_{chip}}{E_b/N_0} \cdot \frac{P_i^{rx}}{(1-\omega)P_i^{rx} + I_{DCH}(n_D) + N} \cdot \frac{1}{n}. \tag{3}$$

Expression (3) shows that $R_i(\underline{n})$ depends on the current state $\underline{n}$ only via $c(\underline{n})$ and n_D. Considering the third step of the analysis, in the special case where $\lambda_D = 0$, the Markov model for the OBO scheduler is effectively a multi-class $M/M/1$ processor sharing model, which is well examined and an explicit expression for the steady-state distribution is available, see e.g. [3].

5.2 Partial Parallel (PP) Scheduler

As the PP scheduler allows parallel transmissions of multiple EDCH calls in a single TTI, $I_{EDCH}(\underline{n})$ comprises the interference contributions from all scheduled EDCH calls in a TTI. Consequently, the instantaneous rate depends on $\underline{n}$, see (1). The opportunity for simultaneous transmissions also results in a shorter cycle than under OBO scheduling which can be expressed as the ratio of the aggregate resource requested by all present EDCH calls and the available EDCH budget $B'(n_D)$. The cycle length is then given by

$$c(\underline{n}) = \max\left\{1, \frac{\sum_{i=1}^{K} n_i P_{i,\max}^{rx}}{B'(n_D)}\right\} \tag{4}$$

and hence the state-dependent throughput $R_i(\underline{n})$ is equal to

$$R_i(\underline{n}) = \frac{r_{chip}}{E_b/N_0} \cdot \frac{P^{rx}_{i,\max}}{B'(n_D) - \omega P^{rx}_{i,\max} + I_{DCH}(n_D) + N} \cdot \frac{B'(n_D)}{\sum_{i=1}^{K} n_i P^{rx}_{i,\max}}, \quad (5)$$

if $\sum_{i=1}^{K} n_i P^{rx}_{i,\max} \geq B'(n_D)$. In the alternative case that the sum of the power levels of the active users is lower than the budget, i.e. if $\sum_{i=1}^{K} n_i P^{rx}_{i,\max} < B'(n_D)$, each active user sends in each TTI and the cycle length is $c(\underline{n}) = 1$. That simplifies the state-dependent throughput expression to

$$R_i(\underline{n}) = \frac{r_{chip}}{E_b/N_0} \cdot \frac{P^{rx}_{i,\max}}{I_{EDCH}(\underline{n}) - \omega P^{rx}_{i,\max} + I_{DCH}(n_D) + N}, \quad (6)$$

where $I_{EDCH}(\underline{n}) = \sum_{i=1}^{K} n_i P^{rx}_{i,\max}$.

5.3 Full Parallel (FP) Scheduler

Under FP scheduling, an active user transmits in each TTI and therefore the cycle length $c(\underline{n})$ is equal to 1 for all states $\underline{n}$. Hence the state-dependent throughput is equal to the instantaneous rate (calculated for the appropriate received power level), i.e. $R_i(\underline{n}) = r_i(\underline{n})$. In the expression for $r_i(\underline{n})$, $I_{EDCH}(\underline{n})$ comprises contributions from all EDCH calls. We distinguish between two cases. In the first case the number of EDCH calls is such that the sum of their (received) maximum powers is lower than the EDCH budget $B'(n_D)$. In that case the EDCH calls use their maximum transmit power and the state-dependent throughput is the same as under PP scheduling, see (6). The second case is when the summed maximum received power from all users is higher than the EDCH budget. Since all users are assigned to transmit in parallel, the transmit power levels have to be decreased such that the summed received powers fit in the EDCH budget. The resulting received power levels P^{rx}_i are derived from the maximum received power via a proportional decrease:

$$P^{rx}_i = \frac{P^{rx}_{i,\max}}{\sum_{i=1}^{K} n_i P^{rx}_{i,\max}} \cdot B'(n_D), \quad (7)$$

so that $I_{EDCH}(\underline{n}) = \sum_{i=1}^{K} n_i P^{rx}_i = B'(n_D)$. Using P^{rx}_i we can rewrite expression (1) for this second case of FP scheduling as

$$R_i(\underline{n}) = r_i(\underline{n}) = \frac{r_{chip}}{E_b/N_0} \cdot \frac{P^{rx}_i}{B'(n_D) - \omega P^{rx}_i + I_{DCH}(n_D) + N}. \quad (8)$$

6 Numerical Results

In this section we present and discuss numerical results on the performance of the three scheduling schemes under various cell load conditions. The comparison of the three scheduling schemes is based on performance measures such as mean file transfer time for the EDCH users and the fairness index.

6.1 Parameter Settings

In the numerical experiments we assume a system chip rate r_{chip} of 3840 kchips/s, a thermal noise level N of -105.66 dBm and a noise rise target η at the base station of 6 dB. From these parameters the total received power budget B can be calculated: $B = \eta \cdot N$. A self-interference of 10% of the own signal is considered, i.e. $\omega = 0.9$. The assumed path loss model is given by $L(d) = 123.2 + 35.2 \log_{10}(d)$ (in dB).

The considered cell is split in $K = 10$ zones[2]. Given an E_b/N_0 target of 1.94 dB for EUL transmissions, a maximum transmission power of $P_{\max}^{tx} = 0.125$ Watt and a worst case interference level (where the received power budget B is fully used), we applied straightforward link budget calculations to determine the zone radii corresponding to a set of ten bit rates between 256 kbit/s (zone 10) and 4096 kbit/s (zone 1). EDCH calls consist of file transfers of mean size $F = 1000$ kbit; the aggregate rate at which new file transfers are initiated is $\lambda = 0.4$ unless stated otherwise.

DCH users are assumed to generate voice calls with requested bit rate of 12.2 kbit/s, an activity factor of 50% and an E_b/N_0 of 5.0 dB. The mean duration τ of the voice calls is 120 seconds. Given a noise rise target of 6 dB, this translates to a P_D of $0.0729 \cdot 10^{-14}$ Watt. A DCH budget of 70% of the totally available channel resource B was used as a default value, implying a maximum of 77 simultaneous speech calls. Given a target blocking probability of 1%, this translates to a supported speech traffic load of about 62 Erlang, and hence $\lambda_D \approx 0.52$ calls/s. The DCH call arrival rate λ_D is always chosen such that the DCH call blocking probability equals 1%. The use of other than default values will be explicitly indicated where applicable.

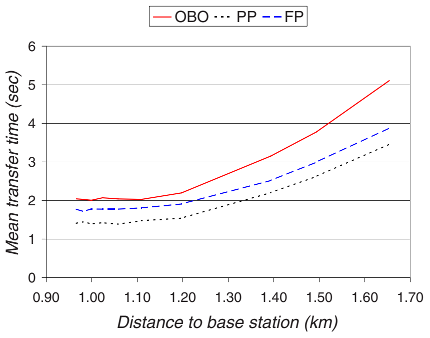
We created a generic simulator in MatLab for deriving the steady-state distribution of the multi-dimensional Markov chain describing the system behaviour at flow level (Step 3 of the analysis approach). This requires relatively short running times. For example, our simulations with confidence intervals of about 1% took typically 2.5 minutes.

6.2 Discussion of Numerical Results

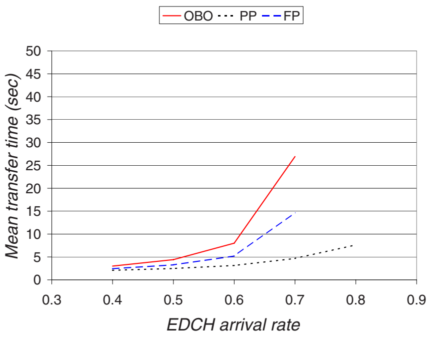
Performance Impact of the User Location

Figure 3(a) shows, for each of the three schedulers, the mean file transfer time as a function of the user's distance from the base station. The system and traffic parameters are set according to their default values indicated above. As expected, the partial parallel (PP) scheduling scheme outperforms the two other schemes, cf. the discussion at the end of Section 2; in the current situation the full parallel (FP) scheme performs second best and the one-by-one (OBO) scheme shows the worst performance. For all three schedulers, when moving away from the base station, the mean flow transfer times remain more or less constant until

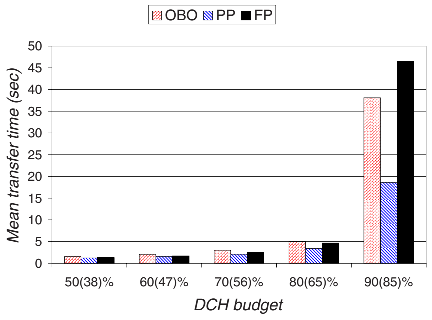
[2] Extensive numerical experiments showed that this granularity is sufficient for our purposes. Finer granularities, e.g. $K = 20$ or $K = 40$, did not provide essentially different results.



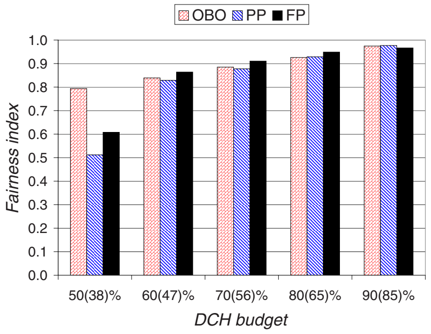
(a) Impact of user location



(b) Impact of EDCH load



(c) Impact of DCH budget (load)



(d) Fairness for different DCH budgets

Fig. 3. Performance comparison of the three scheduling schemes

a distance of about 1.1 km. Apparently, in these central zones the combination of the available budget and the maximum attainable received power allows the same (maximal) data rates. At larger distances the effect of the increasing path loss, consequently lower attainable received powers and hence lower bit rates becomes clearly visible through rapidly increasing flow transfer times.

Note, that even for users close to the base station, which are able to fill up the whole EDCH budget on their own, the PP scheme performs considerably better than the OBO (and FP) scheme. This is due to the fact that the particular disadvantage of OBO for users at the cell edge (who cannot fill the budget on their own, and, hence, waste resources compared to PP) also has a disadvantageous effect on the throughputs obtained by the users at the centre of the cell (as the channel access is fairly shared among all users in the cell).

Performance Impact of the E ective EDCH Load

In Figures 3(b)-3(c) the effective EDCH load is varied in two distinct ways. In Figure 3(b), the aggregate EDCH flow arrival rate λ is varied directly. In Figure 3(c), the available capacity for EDCH transfers is varied (which effectively corresponds with an inverse variation of the EDCH load) by varying the DCH budget and load. As we will see, the key difference between these distinct approaches in varying the effective EDCH load is visible in the relative performance of the OBO and FP schedulers.

In Figure 3(b) the mean flow transfer time (appropriately averaged over all zones in the cell) is given as a function of λ. The other parameters are the same as in Figure 3(a). Observe that the performance difference between the schedulers shown by Figure 3(a) becomes even more pronounced when λ increases. In particular, the mean flow transfer time for the OBO (and also FP) scheme increases very rapidly when λ becomes larger than, say, 0.6 flow initiations/sec, while the system becomes saturated for λ's between 0.7 and 0.8. The growth of the mean flow transfer time under PP scheduling remains moderate. Apparently, the relatively inefficient traffic handling in the OBO and FP schemes leads to a considerable reduction of the cell capacity compared to the PP scheme.

As argued before, and illustrated by Figure 3(a) and Figure 3(b), the PP scheduling scheme performs always better than (or at least as good as) the OBO and FP schemes. It is however interesting to consider how the performance gain of PP over the OBO and FP schedulers depends on the available EDCH budget. In particular, it is expected that when the available EDCH budget is small enough to be filled up by a single user, OBO is more efficient than FP, since it yields lower intra-cell interference, higher signal-to-interference ratios and hence higher bit rates. In order to investigate this we have evaluated the scenario of Figure 3(a) under various DCH budgets/loads affecting the (remaining) EDCH budget available for the EDCH users. More specifically, we have varied the DCH budget between 50% and 90% of the total budget B and in each case determined the DCH arrival rate λ_D such that the DCH traffic experiences a blocking probability of 1%. Besides the available DCH budget, the horizontal axis indicates (between brackets) the (average of the) actually used budget by the DCH calls.

Figure 3(c) shows, for each of the schedulers, the resulting mean EDCH flow transfer time versus the DCH budget/load. Obviously, for all three schedulers, the mean EDCH flow transfer times increase when the DCH load increases, since the resources remaining for EDCH transfer decreases. For small values of the DCH load, the performance of the three scheduling schemes is quite similar, in particular for PP and FP. This is due to the fact that when the number of simultaneously ongoing flow transfers is small (typically when the overall system load is small) PP and FP (and to a lesser extent also OBO) handle them effectively in the same way. For higher DCH budgets/loads the performance gain of PP (compared to FP and OBO) increases, while for the highest considered DCH budget/load, the OBO scheme indeed performs better than FP.

Fairness Issues

Finally, for the same scenarios as consider in Figure 3(c), we investigate in some more detail the fairness of the three schedulers with respect to their performance as observed by users at different locations in the cell, cf. Figure 3(a). The fairness can be defined in different ways. We have used the fairness index applied by e.g. Jain [8], which, in the present context, is defined as $(\sum_{i=1}^{K} D_i)^2/[K\sum_{i=1}^{K}(D_i)^2]$, where D_i denotes the mean flow transfer time for users in zone i, $i = 1, ..., K$.

The maximum value of the fairness index equals 1, which refers to a perfectly fair scenario in which the mean flow transfer times are the same for all zones. The smaller the fairness index the larger the (relative) differences among the mean flow transfer times in the different zones.

Figure 3(d) shows the fairness results for the three schedulers. The general impression is that the fairness performance is more or less the same for the three schemes, see also Figure 3. However, for the case that the DCH budget/load is small (i.e. the available EDCH budget is relatively large) the OBO scheme appears to be significantly more fair than the two other schemes, in particular when compared to the PP scheme. This can be explained as follows. Under PP scheduling, users near the base station (with a high maximum received power) are more likely to be served alone compared to users at the cell edge, which are mostly served in parallel with others (because they are unable to utilise the available EDCH budget on their own) and will consequently experience lower signal-to-interference ratios and hence lower bit rates. This will lead to relatively large differences between the throughputs observed by users close to the base station and users at the cell edge. Under OBO scheduling, all users are scheduled in a one by one fashion, including remote users, which therefore do not suffer from the additional intra-cell interference as would be imposed on them under FP scheduling, thus establishing a greater degree of fairness. Observe that the fairness of the schedulers improves when the DCH budget/load increases (i.e. available EDCH budget decreases). This is due to the fact that when the available EDCH budget becomes smaller, the maximum (received) power that can be achieved by the users (i.e. their mobile equipment) at different locations is less and less a limiting factor for remote users, and hence the disadvantageous effect that remote users suffer from added intra-cell interference vanishes.

7 Conclusions

We have presented a modelling and analysis approach for comparing the flow level performance of three scheduling schemes for UMTS EUL, viz. one-by-one (OBO), partial parallel (PP) and full parallel (FP) scheduling, in a single cell scenario taking into account the users' limited uplink transmission power and interference factors such as thermal noise and intra-cell interference from UMTS R'99 uplink users. This hybrid analytical/simulation approach allows for fast evaluation of mean flow transfer times of users at different cell locations.

The numerical results show that the PP scheduling scheme clearly outperforms the two other schemes. Besides delivering higher user throughputs (i.e. smaller flow transfer times), PP also yields a considerably higher system capacity by exploiting the available channel resources in a more efficient way. The FP scheme performs mostly better than OBO, but when the available channel resource for EUL users is low (e.g. when there are many UMTS R'99 users getting preference over the 'best effort' EUL users) then OBO yields lower flow transfer times than FP. Additionally, we observed that the three schedulers do not differ that much in (un)fairness with respect to the performance experienced by users at different locations in the cell. Only in scenarios where the available EDCH budget is relatively high, the schedulers differ significantly in the established fairness. Although in such cases the PP scheduler appears to be unfairest, it is noted that still all users are best off when compared to OBO and FP.

Currently we are extending our flow level performance modelling and analysis approach to scenarios with multiple cells taking into account mutual interactions due to inter-cell interference. One step further towards a more exhaustive research is including user mobility in the analysis approach. In addition, besides the channel oblivious scheduling schemes studied in the present paper, we will also consider channel aware schedulers in order to investigate their potential for performance enhancement and impact on e.g. fairness.

References

1. 3GPP TS 25.308. High Speed Downlink Packet Access (HSDPA); Overall Description
2. 3GPP TS 25.309. FDD Enhanced Uplink; Overall Description
3. Cohen, J.W.: The multiple phase service network with generalized processor sharing. In: Acta Informatica, vol. 12, pp. 254–284 (1979)
4. Fodor, G., Telek, M.: Performance analysis of the uplink of a CDMA cell supporting elastic services. In: Boutaba, R., Almeroth, K.C., Puigjaner, R., Shen, S., Black, J.P. (eds.) NETWORKING 2005. LNCS, vol. 3462, Springer, Heidelberg (2005)
5. Helmersson, K.W., Englund, E., Edvardsson, M., Edholm, C., Parkvall, S., Samuelsson, M., Wang, Y.-P.E., Cheng, J.-F.: System performance of WCDMA enhanced uplink. In: IEEE VTC 2005 (Spring), Stockholm, Sweden (2005)
6. Holma, H., Toskala, A.: WCDMA for UMTS. John Wiley & Sons Ltd., Chichester (2001)
7. Holma, H., Toskala, A.: HSDPA/HSUPA for UMTS. John Wiley & Sons Ltd., Chichester (2006)
8. Jain, R.: The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling. John Wiley & Sons Ltd., Chichester (1991)
9. Kumaran, K., Qian, L.: Uplink scheduling in CDMA packet-data systems. In: Wireless Networks, vol. 12, pp. 33–43 (2006)
10. Li, C., Papavassilious, S.: On the fairness and throughput trade-off of multi-user uplink scheduling in WCDMA systems. In: IEEE VTC 2005 (Fall), Dallas, USA (2005)
11. Mäder, A., Staehle, D.: An analytical model for best-effort traffic over the UMTS enhanced uplink. In: IEEE VTC 2006 (Fall), Montreal, Canada (2006)
12. Ramakrishna, S., Holtzman, J.M.: A scheme for throughput maximization in a dual-class CDMA system. In: ICUPC 1997, San Diego, USA (1997)
13. Rosa, C., Outes, J., Sorensen, T.B., Wigard, J., Mogensen, P.E.: Combined time and code division scheduling for enhanced uplink packet access in WCDMA. In: IEEE VTC 2004 (Fall), Los Angeles, USA (2004)

A Hybrid Fault-Tolerant Algorithm for MPLS Networks

Maria Hadjiona, Chryssis Georgiou, Maria Papa, and Vasos Vassiliou

Department of Computer Science, University of Cyprus, CY – 1678, Nicosia, Cyprus
{cs02cm1,chryssis,cs03pm,vasosv}@cs.ucy.ac.cy

Abstract. In this paper we present a new fault tolerant, path maintaining, algorithm for use in MPLS based networks. The novelty of the algorithm lies upon the fact that it is the first to employ both path restoration mechanisms typically used in MPLS networks: protection switching and dynamic path rerouting. In addition, it is the first algorithm to adequately satisfy all four criteria which we consider very important for the performance of the restoration mechanisms in MPLS networks: fault recovery time, packet loss, packet reordering and tolerance of multiple faults. Simulation results indicate the performance advantages of the proposed hybrid algorithm (with respect to the four criteria), when compared with other algorithms that employ only one of the two restoration mechanisms.

Keywords: MPLS, fault tolerance, algorithms, rerouting, protection switching.

1 Introduction

The explosive increase of data circulation over the Internet in conjunction with the complexity of the provided Internet services have negatively affected the quality of service and the data flow over this global infrastructure. The Multi-Protocol Label Switching (MPLS) [21] combines the scalability of the IP protocol and the efficiency of label switching to improve network data circulation.

The protection of data flows in the case of link or router failures is very important, especially for real time services and multimedia applications. MPLS employs two basic techniques for network recovery: (i) protection switching, where a pre-computed alternative path, which is usually disjoint from the working path, is set up for every flow and (ii) rerouting, where an alternative path is dynamically recomputed after a fault is detected. For both techniques, the alternative path can be either global or local [22].

The recovery of the MPLS network is based on the algorithm that is applied in order to detect the faults and route the data flow in an alternative path. There are various algorithms that have been proposed in the bibliography. However, each algorithm employs only one of the two basic techniques.

The main motivation for this work is to overcome the drawbacks of the previously proposed schemes for the restoration mechanism in MPLS networks during link/node failure. As already mentioned, existing algorithms use either rerouting [2, 6, 8, 10, 13, 14, 16, 23] or protection switching [1, 3, 4, 5, 7, 10, 12, 17, 18, 20] to reroute traffic fast when a fault occurs in the MPLS domain. Each technique presents both advantages and disadvantages depending on the application or the topology of the network they are employed upon. Protection switching provides fast restoration when compared to the rerouting technique, since the alternative path is already established and the switching to it is performed immediately after the fault is detected. Alternatively, the rerouting technique appears to be better in handling multiple faults, since a new alternative path, if needed, is computed dynamically for each fault. A question driven by the comparison of the two techniques is whether the combination of rerouting and protection switching will give better results.

We consider fault recovery time, packet loss, packet reordering and the ability to tolerate multiple faults as the most important criteria to evaluate a fast restoration algorithm in MPLS networks. To the best of our knowledge there is no current algorithm, either protection switching or rerouting, able to perform well in all four performance criteria. The challenge is to find an efficient way to combine the two restoration mechanisms in order to exploit each method's strengths and obtain a new hybrid algorithm that would perform best in all four criteria.

In this paper we propose and evaluate such a hybrid fault-tolerant path-maintaining algorithm for use in MPLS based networks. It satisfies all four abovementioned performance criteria and deploys effectively, in a non-trivial manner, both mechanisms based on the conditions of the fault, thus exploiting the advantages of each technique. Simulation results demonstrate the effectiveness of the new approach.

2 Related Work

2.1 Performance Criteria

Several criteria to compare the performance between different MPLS-based recovery schemes are defined in [20]. These are: packet loss, additive latency, re-ordering, recovery time, full restoration time, vulnerability, and quality of protection. Fault recovery time is the time elapsed between the fault detection and the time when the first packets are rerouted using the alternative path. Recovery time includes additive latency and sometimes is the equivalent to full restoration time. Packet loss is the percentage of packets lost until the fault is recovered. Packet reordering is whether the packets delivered during the recovery period are delivered out-of-order or not. Vulnerability is the time that the protected LSP (Label Switching Path) is left unprotected and quality of protection is the probability of a connection to survive the failure.

For the purposes of our work, which focuses more on the global performance and fault-tolerance of the MPLS network, we consider the first three criteria (recovery time, packet loss and packet disordering) and instead of vulnerability and quality of protection, we consider, as a more suitable fourth criterion, the ability of the network to tolerate multiple faults.

2.2 Comparison

Several service restoration algorithms are proposed for MPLS networks, each employing one of the two restoration mechanisms. Particularly, protection switching technique is employed by Haskin [12], Makam [17], Gonfa [10], Two Path [3, 4, 5], RBPC [1], Dual [7], MBAK [20], and SPM [18] algorithms. On the other hand, rerouting technique is employed by Dynamic Routing [10], A.J.C [2], Yoon [2], [23], Chen & Oh [2, 8], Otel [19], MIRA [6, 14], Hongs [13], and Lin & Lui [16] algorithms. We studied each of these algorithms and evaluated them (in a theoretical manner) based on the abovementioned four criteria.

The characterization and evaluation of the existing fault tolerance algorithms based on the selected criteria is shown in Table 1. The table is divided with two horizontal parts; the upper level contains protection switching algorithms, whereas the lower level contains rerouting algorithms. The symbol "+" indicates that the specific algorithm satisfies adequately the corresponding criterion. Due to lack of space we do not present the justification of whether an algorithm adequately satisfies a criterion or not, but the interested reader can obtain this information along with a detailed description of each algorithm in [11].

One can observe that the two algorithms which satisfy the most criteria are Gonfa [10] and Otel [19]. The Gonfa algorithm is a protection switching algorithm which performs well with respect to recovery time, packet loss and packet reordering criteria. On the other hand, Otel algorithm is a rerouting algorithm which performs well with respect to recovery time, packet loss and tolerance of multiple faults. In addition it can be observed that the two algorithms are not able to satisfy all four criteria (only three each). Based on these observations we decided to develop a new algorithm that makes use of these two algorithms and is able to satisfy adequately all four criteria. The new algorithm is presented in the next section.

Table 1. Comparison of existing algorithms based on the four selected criteria

Algorithms	Local(L), Global(G) Restoration	Recovery Time	Packet Loss	Packet Reordering	Multiple Faults Tolerance
Makam	G			+	
RBPC (Local)	L				+
Two Path	L				+
MBAK	G				+
RBPC (Global)	G			+	+
Dual	L	+	+		
Haskin	L	+	+		
Gonfa	L,G	+	+	+	
Dynamic Routing	L				+
Hongs	L				+
MIRA	G			+	+
Lin & Lui	G			+	+
A.J.C	L			+	+
Chen & Oh	L	+	+		
Yoon	L	+	+		
Otel	L	+	+		+

3 The Hybrid Algorithm

In this section we give a brief description of the new algorithm, referred as Hybrid, and describe its execution through an example.

3.1 Description of the Algorithm

The hybrid algorithm maintains four data structures: (i) a Shortest Path Tree (SPT) where the root is the node that will execute the calculations, (ii) an array of lengths which contains the length of the shortest paths between the SPT root and all other nodes, (iii) a priority queue for nodes, (iv) a list maintained by ingress LSR (Label Switching Router) and contains the working and alternative LSP. The first three data structures are the ones also used by the Otel algorithm (details in [19] or [11]). Hence, the additional state information compared to [19] is the fourth structure.

First, the establishment of the working LSP and the alternative LSP which protects the whole working LSP is fulfilled. Afterwards the segment protection domains are determined along with their backward LSPs. Then the backward LSPs are established. The alternative and backward LSPs are established based on the Gonfa algorithm.

The alternative LSP and backward LSPs are used by data flows with low priority. (When a fault occurs, the LSPs will be needed for restoration of the fault. Hence the low priority flows will stop routing via those paths in order to forward the influence data flow with high priority). In addition, all the abovementioned data structures are created. Once the initialization phase is complete, the algorithm begins its path maintenance operation using the Gonfa algorithm. Depending on the nature and location of a fault as well as the current state of the network topology, the algorithm might divert in using the Otel algorithm and back (along with some additional calculations), as can be observed by the Hybrid algorithm's outline, given in Figure 1. The data structure SPT is updated with the use of any Single Source Shortest Part algorithm (SSSP) [15]. Full details of the hybrid algorithm can be found in [11].

3.2 Example of an Execution of the Algorithm

For a better understanding of the algorithm we describe a specific execution of the algorithm and make use of the network topology given in Figure 2. The working path is established between LSR1, LSR3, LSR5, LSR7, LSR9 and LSR11 and the alternative path is established between LSR1, LSR2, LSR4, LSR6, LSR8, LSR10 and LSR11. The first backward LSP is LSR3, LSR1, the second backward LSP is through LSR7, LSR5, LSR3 and the third backward LSP is formed by LSR11, LSR9 and LSR7.

The first link failure occurs between LSR5-LSR7 and based on the Hybrid algorithm's description the fault is recovered using the Gonfa algorithm. The data flow is routed via the backward LSP which it is formed by LSR5 and LSR3 and then follows the alternative path LSR4, LSR6, LSR8, LSR10 και LSR11. Afterwards the traffic is routed directly to the alternative path by LSR3, as shown in Figure 2.

```
Begin:
Establish working, alternative and backward LSPs
Compute:
  1.  SPT
  2.  Array of lengths
  3.  Array with the pre-established paths in Ingress LSR
Set _working_LSP as available
Set _alternative_LSP as available
Run the Gonfa algorithm

When failure occurs check:
If (failure is in working path)
        Set _working_LSP as NOT available
If (failure is in alternative path)
        Set _alternative_LSP as NOT available
If ( _working_LSP IS available && _alternative_LSP IS available)
        Update SPT using SSSP and array of lengths
If ( _working_LSP IS NOT available && _alternative_LSP IS available)
        Step1: Recover from fault using the Gonfa algorithm
        Step2: Update SPT using SSSP and array of lengths
If ( _working_LSP IS available && _alternative_LSP IS NOT available)
        Update SPT using SSSP and array of lengths
If (_working_LSP IS NOT available && _alternative_LSP IS NOT available)
        Recover from fault using the Otel algorithm

When repair of a failure occurs check:
Step1: Update SPT using SSSP and array of lengths
Step2: Check: working LSP is repaired?
If (compare array with pre-established paths in Ingress LSR and SPT)
        Step1: Reroute the traffic in the working LSP
        Step2: Set _working_LSP as available
Else (compare array with alternative paths in Ingress LSR and SPT)
        Set _alternative_LSP as available
```

Fig. 1. Outline of the Hybrid Algorithm

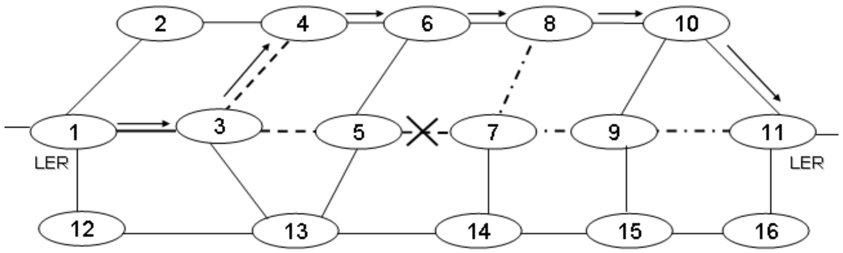


Fig. 2. Recovery from the fault using the Gonfa algorithm

The next step is to update all data structures of every node. In this example we concentrate on LSR3, as LSR3 is the upstream LSR of the next fault and consequently LRS3 will become responsible in finding the alternative LSP and route the flow. In Figure 3(a) the SPT before the failure is shown and in Figure 3(b) the SPT after the failure is shown.

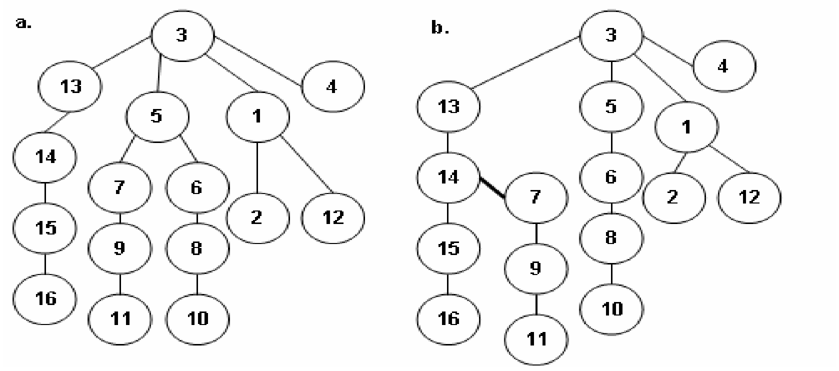


Fig. 3. (a) SPT before link failure. (b) SPT after link failure

The next fault occurs on the link connecting LSR3 and LSR4. Per the Hybrid algorithm's description the fault can be repaired with the use of the Otel algorithm. The SPT of LSR3 after the first fault is shown in Figure 3(b). Hence the next step is to consider the SPT subtree rooted at the disconnected downstream LSR and then starting with the subtree root, all the nodes in this subtree are marked as "unreachable". In this case the unreachable node is only LSR4 and the destination node is a reachable node. Therefore there is a path which can be used to route the data to the destination node. The local path is LSR3, LSR13, LSR14, LSR7, LSR9, and LSR11. Figure 4 shows how the data flow is routed after the second failure.

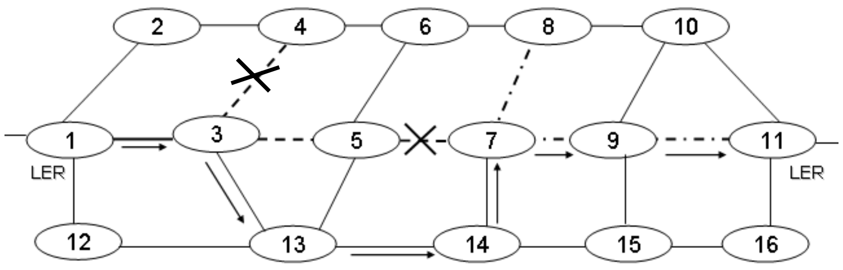


Fig. 4. Recovery from the second fault

After routing the flow via the local alternative LSP, the Otel algorithm continues in order to update its data structures. The SPT is updated by deleting the branch linking LSR4 and its parent LSR3 and adding as an SPT branch the linking LSR2 and LSR4, as shown in Figure 5.

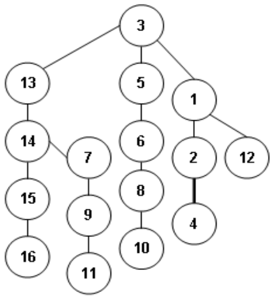


Fig. 5. SPT after second link failure

4 Simulations and Analysis

We have performed simulations that experimentally evaluate and contrast the performance (based on the four criteria) of the Hybrid algorithm with the performance of Otel and Gonfa (i.e., the best representatives of each restoration mechanism). The simulations were performed on ns2 [9]. We experimented using several different simulation scenarios, considering different network topologies (including mesh, star, hierarchical with single- and multi-homed nodes, and common telecommunication networks in USA and UK). The simulation results were analyzed by grouping the topologies in categories depending on the number of nodes and links they have. We call a topology with a small number of nodes (<10) as *simple*; otherwise we call it *complex*. Also, depending on the density of the topology (small/large number of links) it is called *sparse* or *dense* (per the standard graph-theoretic definitions). Hence, we have four different topology categories: simple and sparse, simple and dense, complex and sparse, complex and dense. Due to lack of space, here we present and analyze simulation results for the category of *complex and dense* topologies (which includes the hierarchical multi-homed and the USA telecommunication network topologies). Readers are referred to [11] for further details and for the results of other categories.

The evaluation of the three algorithms was based on different scenarios. The type and order of the faults were different in each simulated case so to cover different events that may occur at any time. All scenarios include multiple (two) faults. Here, we present and examine the following four scenarios:

Scenario 1: The first fault occurs on the working path and the second fault occurs on Gonfa's alternative path (which is different from the Otel's alternative path).
Scenario 2: The first fault occurs on the working path and the second fault occurs on Otel's alternative path (which is different from the Gonfa's alternative path).
Scenario 3: The first fault occurs on the working path and the second fault occurs on a common link for the two alternative paths (Otel and Gonfa).
Scenario 4: The first fault occurs on Gonfa's and Hybrid's pre-established alternative path and the second fault occurs on the working path.

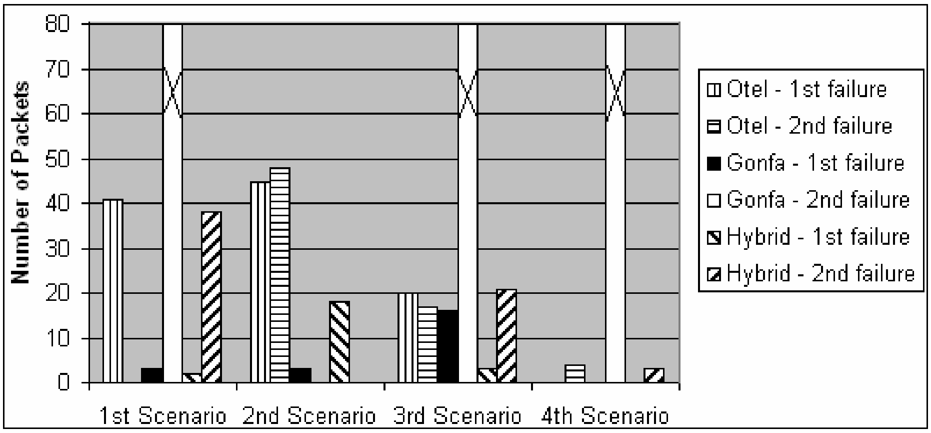


Fig. 6. Packet loss in complex and dense topology

Figure 6 presents the results for packet loss. The Hybrid algorithm is able to recover from faults with lower packet loss compared to the other two algorithms in all but one of the scenarios. The 1st and 3rd scenarios are the scenarios with the highest packet loss for the Hybrid algorithm. This is mainly due to the fact that the topology is complex and dense and the SPT is much larger, thus more time is needed to calculate an alternative path. As expected, the Gonfa algorithm fails to recover traffic from the failures where both working and alternative paths were affected. In those cases, enormous packet loss was observed (as the algorithm cannot deliver packets anymore), which is shown by a cross in the respective chart bar. Also high packet loss is observed for the Otel algorithm in the two cases where the faults occur both in the working and the alternative path and the algorithm must reroute the traffic twice.

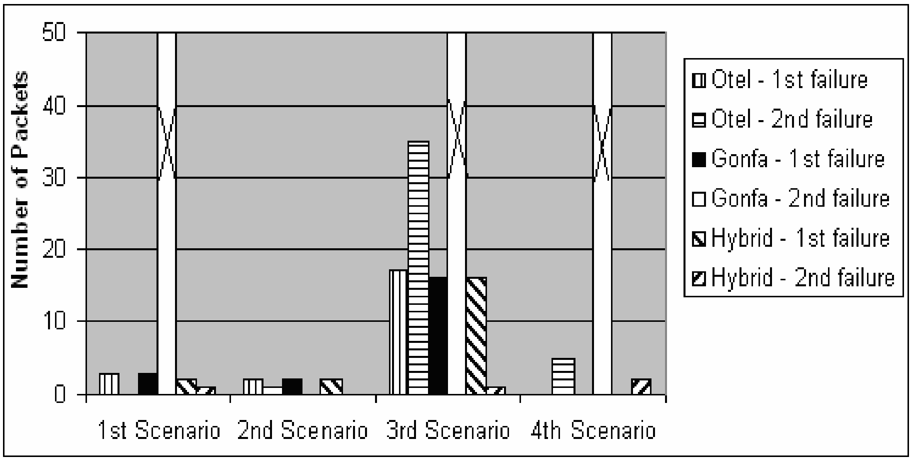


Fig. 7. Packet reordering in complex and dense topology

In Figure 7 the number of re-ordered packets is shown. Generally, packet reordering is approximately the same for the three algorithms, in cases where all three algorithms are able to reroute the traffic. There is a small increase on packet reordering in the 3rd scenario in all algorithms. Once more we experience high packet reordering for the Gonfa algorithm in the 1st, 3rd and 4th scenarios due to its failure to recover from the second fault. In all cases the number of re-ordered packets for the hybrid algorithm is less or at least the same with the lowest numbers in the other two algorithms.

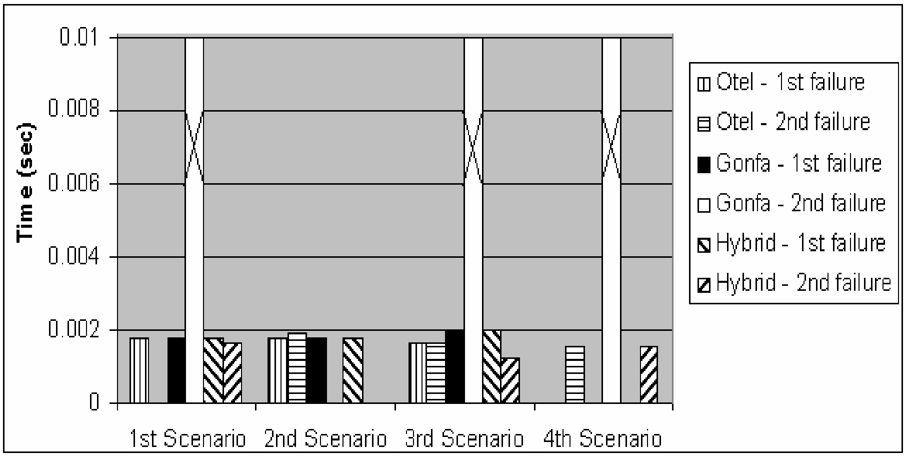


Fig. 8. Fault recovery time in complex and dense topology

The recovery time for each fault is shown in Figure 8. In the first scenario all three algorithms are able to recover from the first failure, as expected. In the second failure the Otel algorithm is not influenced and its recovery time is zero. In contrast, the hybrid and Gonfa algorithms are affected by the failure. The Hybrid algorithm is able to find an alternative path in 0,0016 seconds. However the Gonfa algorithm was not able to recover traffic from the failure (indicated by a cross in its chart bar).

In the second scenario it is observed that all algorithms are able to recover from the first failure with the same recovery time. The second failure only influences the Otel algorithm as the fault occurs on its alternative path; the algorithm is able to recover from the failure in 0,0019 seconds. The Gonfa and Hybrid algorithms are not affected by the fault and the recovery time is therefore zero.

Once again in the third scenario, all algorithms are able to recover from the first failure. As for the second fault, the Gonfa algorithm was not able to recover traffic from the failure. The Otel algorithm needs 0,0016 seconds to recover from the second failure whereas the Hybrid algorithm needs 0,0012 seconds.

In the last scenario of the complex and dense topology, the first fault does not influence the data flow of the algorithms as it is occurred on Gonfa's pre-established alternative path (i.e., the working path is unaffected). For this reason the recovery time for the first fault is zero for all three algorithms. As for the second fault, Gonfa fails to recover. On the contrary, both Otel and Hybrid are able to calculate dynamically an alternative path and switch the traffic in the alternative path in 0,0015 seconds.

Based on the obtained simulation results, we draw the following conclusions regarding the proposed Hybrid algorithm. First of all, each topology category gives, not surprisingly, different results for the same failure scenario. The network topology plays a significant role in the Hybrid algorithm (and Otel) due to the fact that the alterative path is calculated via the SPT, which basically represents the topology. Based on the obtained results a small increase on packet loss, packet reordering and the recovery time is observed while the network topology is becoming more complex and dense. This is due to the fact that when the SPT becomes bigger, it requires more time to be updated (for the computation of a new alternative path). Moreover it appears that the performance of the algorithm depends on two correlated factors: (i) the size of the SPT subtree, affected by the failure and (ii) the number of links originating from nodes outside the subtree but incident to subtree nodes.

Furthermore, two different SPTs can give different results in the same scenarios. This is derived from the different recovery times given by the Otel and Hybrid algorithms in the same scenario. The hybrid algorithm uses SSSP to update the SPT each time a fault occurs and does not affect the flow of the data. This procedure is not included in the Otel algorithm, hence different SPTs are developed for the two algorithms. As explained before, the structure of the SPT is a basic factor for the performance. Therefore different recovery times can be observed.

The Hybrid algorithm can approach the same recovery time as the Gonfa algorithm when the former is at the stage where it employs the protection switching technique. The cases where the Gonfa algorithm is called to restore the flow are when single and multiple faults occur in the working path and the alternative path is still available. In addition, these cases are considered to be the ideal cases for the Hybrid algorithm, since the protection switching technique provides fast restoration of the flow. Moreover, the hybrid algorithm can approach the same recovery time as of Otel's, in the case of repairing a fault using the rerouting technique. (The Otel algorithm is considered to be one of the best rerouting algorithms with respect to fault recovery time). Sometimes it is observed that in the same scenario the two algorithms may have different fault recovery times. The reason is that the two algorithms may develop different SPTs and this leads to different recovery times. Nevertheless, when the same SPT is developed for both algorithms the same recovery times is measured (as expected).

Per the simulation results, high packet loss is observed in the Hybrid algorithm when the Otel algorithm is called to restore the flow, especially when the topology is complex and dense. When the topology is simple, low packet loss is observed, since the SPT is simpler and requires less time to calculate an alternative path. The lowest packet loss is given when the Gonfa algorithm is applied in order to reroute the traffic to a pre-established path.

As for the packet reordering, the simulations have shown that the number of packets received in out-of-order is relatively small in most scenarios. While the topology is becoming more complex and dense, a continued increase in packet reordering is observed, especially when the Otel algorithm is applied. When the Gonfa algorithm is followed packet reordering is low.

It is evident from the simulation results that the Hybrid algorithm can tolerate multiple faults in the working path as well as in the alternative paths (as it can employ dynamic rerouting) regardless of the topology category (and provided of course that an alternative path exists). To conclude, it appears that the Hybrid algorithm combines effectively the advantages of protection switching and dynamic rerouting restoration techniques and hence it is able to reroute the traffic as many times as the number of failures detected (if needed to do so).

5 Conclusions

The algorithm presented in this paper is the first hybrid algorithm that employs both protection switching and path rerouting restoration mechanisms in an effort to perform well in a number of important criteria. The Hybrid algorithm combines effectively both mechanisms and decreases the fault recovery time, reduces the packet loss and packet reordering in several cases (when compared with algorithms that employ only one of the two mechanisms) and supports multiple link and node failures both on the working and recovery paths.

For future work we plan to enhance the Hybrid algorithm with Quality of Service criteria in its path selection process. This will enable us to provide combined fault tolerance and traffic engineering in VC-based networks.

References

1. Afek, Y., Bremler-Barr, A., Kaplan, H., Cohen, E., Merritt, M.: Restoration by Path Concatenation: Fast Recovery of MPLS Paths. In: Proceedings of the twentieth annual ACM symposium on Principles of Distributed Computing, pp. 43–52 (2001)
2. Ahn, G., Jang, J., Chun, W.: An Efficient Rerouting Scheme for MPLS-Based Recovery and Its Performance Evaluation. Telecommunication Systems 19(3-4), 481–495 (2002)
3. Bartos, R., Raman, M.: A Heuristic Approach to Service Restoration in MPLS Networks. In: Proceeding of ICC 2001, pp. 117–121 (2001)
4. Bartos, R., Raman, M.: A Scheme for Fast Restoration in MPLS Networks. In: Proceedings of the Twelfth IASTED International Conference on Parallel and Distributed Computing Systems (PDSC), pp. 488–493 (November 2000)
5. Bartos, R., Raman, M., Gandhi, A.: New Approaches to Service Restoration in MPLS-Based networks. In: Proceedings of EUROCON 2001 International Conference on Trends in Communications, pp. 58–61 (2001)
6. Capone, A., Fratta, L., Martignon, F.: Dynamic Routing of Bandwidth Guaranteed Connections in MPLS Networks. Wireless and Optical Communications 1(1), 75–86 (2003)
7. Chen, J., Chiou, C.C., Wu, S.L.: A Fast Path Recovery Mechanism for MPLS Networks. In: Kim, C. (ed.) ICOIN 2005. LNCS, vol. 3391, pp. 58–65. Springer, Heidelberg (2005)
8. Chen, T.M., Oh, T.H.: Reliable Services in MPLS. IEEE Communications Magazine 37, 58–62 (1999)
9. Fall, K., Varadhan, K.: The network simulator -ns-2. The VINT project. UC Berkeley, LBL, USC/ISI, and Xerox PARC, `http://www.isi.edu/nsnam/ns/`
10. Gonfa, L.H.: Enhanced Fast Rerouting Mechanisms for Protected Traffic in MPLS Networks. Phd Thesis, Technical University of Catalonia (February 2003)
11. Hadjiona, M., Georgiou, C., Papa, M., Vassiliou, V.: A Hybrid Fault-Tolerant Algorithm for MPLS Networks, Technical Report TR-07-06, Department of Computer Science, University of Cyprus (December 2007),
`http://www.cs.ucy.ac.cy/~chryssis/MPLS-TR.pdf`

12. Haskin, D., Krishnan, R.: A Method for Setting an Alternative Label Switched Paths to Handle Fast Reroute. Internet Draft (2000)
13. Hong, D.W., Hong, C.S.: A Rerouting Scheme with Dynamic Control of Restoration Scope for Survivable MPLS Network. In: Kim, C. (ed.) ICOIN 2005. LNCS, vol. 3391, pp. 233–243. Springer, Heidelberg (2005)
14. Kodialam, M., Lakshman, T.V.: Minimum Interference Routing with Applications to MPLS Traffic Engineering. In: Proceedings of IEEE Infocom, pp. 884–893 (2000)
15. Kurose, J.F., Ross, K.W.: Computer Networking – A Top-Down Approach Featuring the Internet, 3rd edn. Addison-Wesley, Reading (2004)
16. Lin, J.W., Liu, H.Y.: An Efficient Fault-Tolerant Approach for MPLS Network Systems. In: Cao, J., Yang, L.T., Guo, M., Lau, F. (eds.) ISPA 2004. LNCS, vol. 3358, pp. 815–824. Springer, Heidelberg (2004)
17. Makam, S., Sharma, V., Owens, K., Huang, C.: Protection/Restoration Mechanism for MPLS Networks. Internet Draft (1999)
18. Menth, M., Milbrandt, J., Reifert, A.: Self-Protecting Multipaths - A Simple and Resource-Efficient Protection Switching Mechanism for MPLS Networks. In: Proceeding of IFIP-TC6 Networking Conference (Networking), Athens, Greece, pp. 526–537 (May 2004)
19. Otel, F.D.: On Fast Computing Bypass Tunnel Routes in MPLS-Based Local Restoration. In: Proceedings of 5th IEEE International Conference on High Speed Networks and Multimedia Communications, pp. 234–238 (2002)
20. Pu, J., Manning, E., Shoja, G.C.: Reliable Routing in MPLS Networks. In: Proceedings of IASTED International Conference on Communications and Computer Networks (CCN 2002) (2002)
21. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (2001)
22. Sharma, V., Hellstrand, F.: Framework for Multi-Protocol Label Switching (MPLS)-based Recovery. RFC 3469 (2003)
23. Yoon, S., Lee, H., Choi, D., Kim, Y., Lee, G., Lee, M.: An Efficient Recovery Mechanism for MPLS-based Protection LSP. In: Proceedings of Joint 4th IEEE International Conference on ATM (ICATM 2001) and High Speed Intelligent Internet Symposium, pp. 75–79 (2001)

Reliable Signalling Transport in Next Generation Networks

E. Vázquez, M. Álvarez-Campana, A. Hernández, and J. Vinyes

ETS Ing. Telecomunicación, Universidad Politécnica de Madrid
Av. Complutense s/n, 28040, Madrid, Spain
{enrique,mac,albertoh,vinyes}@dit.upm.es

Abstract. Next-generation network architectures aim at the convergence of services (e.g. telephony, television, Web access, online games, and new services) over a common IP core. Additionally, the integration of wired and wireless access technologies will offer users ubiquitous access to all those services. Telecommunications networks rely on signalling protocols to perform key service control functions such as locating and authenticating users, establishing communication sessions, reserving resources, charging, etc. Signalling is crucial for network operators and so requires a transport service with very high availability and low delay. Traditionally, operators have deployed dedicated and highly redundant signalling networks following ITU-T standards. More recently, and motivated by the current evolution towards IP, the IETF has defined SIGTRAN, a new protocol suite intended to transport signalling protocols over IP. This paper addresses the problem of configuring the protocol mechanisms existing at different layers of the SIGTRAN architecture in a coordinated way, with the goal of creating a redundant network topology able to meet the stringent availability levels required for signalling in carrier-grade networks. The proposed solution is described and evaluated numerically.

Keywords: Signalling, SIGTRAN, IP, network availability.

1 Introduction

The fixed telephone network and mobile networks such as GSM, GPRS, and UMTS employ the Signalling System no. 7 (SS7) standardized for international use by ITU-T [1]. SS7 is a packet-switched network designed specifically to transport call control messages (e.g. call setup, call release) among telephone exchanges with low loss, low delay, and very high availability. An SS7 network exists in parallel to the telephone network to which it serves. A few trunk circuits are dedicated to carrying signalling messages, and the remaining circuits carry user calls.

Over the years, the SS7 architecture was successfully adapted to support Intelligent Network (IN) services, broadband networks, and 2G/3G mobile networks [2]. In this evolution process, new types of nodes have been connected to the SS7 networks in addition to telephone exchanges, for instance computers that provide IN services, databases that store information about mobile subscribers, and radio base station controllers. Also, new signalling protocols have been added in the upper layers of the SS7 architecture to communicate with the novel elements. These protocols implement transaction control, mobility management, security procedures, and others that greatly extend the basic call control functionality of the original SS7. On the contrary, the lower layers of the SS7 architecture, which are responsible for the reliable transfer of signalling messages, have remained essentially unchanged.

Recently, the IETF defined the architecture for signalling transport over IP networks known as SIGTRAN [3]. SIGTRAN includes several adaptation protocols and a common transport which normally is the new Stream Control Transmission Protocol (SCTP) [4]. See Fig. 1. In applications where high availability is not required, TCP may replace SCTP. (Also, SCTP is increasingly used for other purposes, different from signalling transport.) Depending on the adaptation chosen, SIGTRAN can transport different signalling protocols.

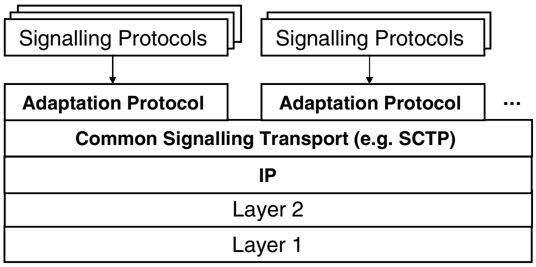


Fig. 1. Signalling Transport (SIGTRAN) over IP

A typical usage of SIGTRAN is in Signalling Gateways that allow new voice over IP networks to interwork with traditional telephone networks using circuit switching and SS7. The gateway receives the signalling messages coming from the SS7 network and forwards them on the IP side with SIGTRAN. In the opposite direction, the messages received from the IP side are forwarded by the gateway to the SS7 network.

SIGTRAN can also be used by fixed and mobile network operators as one step in the evolution to a common IP-based transport infrastructure, not only for user traffic but also for signalling. Mobility control messages increase the signalling traffic in mobile networks compared to fixed ones. Moreover, in GSM the enormous success of the Short Message Service (SMS) is another important factor that raises the signalling network load, because short messages are transported over SS7. In this situation, SIGTRAN can replace parts of the SS7 network in order to offload traffic to IP. This can be done at layer 2 or at layer 3. In the first case, reliable SCTP connections over IP (called *associations*) replace selected links in an SS7 network. The adaptation protocol used over the SCTP association provides the service interface expected by the SS7 network layer, which remains unchanged. The SS7 routing and management procedures are kept to provide network monitoring and recovery from failures.

This paper focuses on the second case, in which the three lower layers of the SS7 network, called the Message Transfer Part (MTP), are completely replaced by an IP network with SIGTRAN. The end-to-end SS7 protocols located above the MTP do not change. Inside the network, signalling traffic is carried over IP, while SIGTRAN/MTP gateways provide connectivity to external SS7 networks. This approach has important advantages for the operator. Network nodes such as telephone switches, user databases, and short message service centers send signalling and SMS traffic over IP interfaces. The MTP protocols are required in the gateways only. If necessary, the transport capacity available for signalling or SMS can be increased in a more flexible and economical way than in a traditional SS7 network using dedicated circuits. Additionally, for a mobile network operator the adoption of SIGTRAN fits well in the evolution path to All-IP 3G/4G networks.

However, the replacement of the Message Transfer Part in a real SS7 network is not trivial. The SIGTRAN protocols and the underlying IP network have to be carefully configured in order to maintain the quality of service, security and, most important, the high availability required by signalling. This is the problem addressed here.

Both SCTP and the adaptation protocols include mechanisms designed to handle redundant configurations and improve availability, most notably SCTP multi-homing, which protects against network failures, and support for active/backup server processes at the adaptation layer, which protects against node failures. While multi-homing and other SCTP features have been widely studied, there are comparatively few references in the literature about configuration of the adaptation protocol.

The next section gives necessary background on SS7 networks, focusing on the functions that are more relevant for this work. The contributions of the paper are presented in sections 3 and 4. Section 3 analyzes the transport and adaptation functions that increase the SIGTRAN network availability, and then defines a configuration of SIGTRAN gateways with multi-homed associations and redundant servers which can be applied to migrate from SS7 to SIGTRAN. Section 4 evaluates the load in the proposed SIGTRAN network. To conclude, section 5 reviews existing related work, and section 6 summarizes the contributions of the paper.

2 Overview of Signalling System no. 7

2.1 Architecture

As mentioned in the introduction, Signalling System no. 7 is an ITU-T standard [1] that defines a packet-switched network and its corresponding protocols for transporting signalling information among the different elements (telephone exchanges, computers, data bases, etc.) of a telephone network. The end nodes making use of the SS7 network services are called Signalling Points (SP), and the intermediate switching nodes within the SS7 network are called Signalling Transfer Points (STP). See Fig. 2. The connections among them are realized by means of Signalling Links, which are usually based on 64 kbit/s channels of E1 or T1 lines.

To cope with node or link failures, redundant network topologies are employed, so that there are several alternatives for routing messages towards any particular destination point. Adjacent SS7 points are usually connected by a Link Set formed by two or more signalling links. Furthermore, end points are attached to two different transfer points, and these are interconnected in a full-meshed topology. Typically, the operator of a large SS7 network will divide it into several regions, each of them having a mated pair of transfer points that serve the signalling points in the region, and will interconnect the different regions with a full mesh of signalling links.

Fig. 2 shows the SS7 protocol architecture. The Message Transfer Part (MTP) provides a basic datagram-like network service for the exchange of messages between signalling points. Further details of MTP are given in section 2.2. The Signalling Connection Control Part (SCCP) provides additional service capabilities such as connection-oriented transfer, segmentation and reassembly, and extended addressing.

The upper SS7 components can be classified into two categories: User Parts and Application Parts. User Parts are traditional signalling protocols oriented to establish, maintain, and terminate calls. The Integrated Services User Part (ISUP) is the standard for establishing and releasing calls in telephone networks, both fixed and mobile.

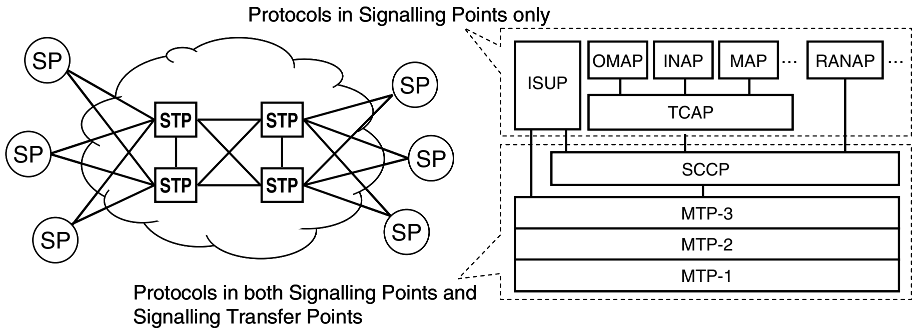

Protocols in Signalling Points only
SP
STP
ISUP
OMAP
INAP
MAP
RANAP
TCAP
SCCP
MTP-3
MTP-2
MTP-1
Protocols in both Signalling Points and Signalling Transfer Points

Fig. 2. SS7 Network and Protocol Architecture

Application Parts perform an increasing number of functions which divert from basic circuit/call control, for example the Intelligent Network Application Part (INAP), the Mobile Application Part (MAP), and the Operation and Maintenance Application Part (OMAP). These elements follow a client-server model, based on remote operations supported by the Transaction Capabilities Application Part (TCAP). Radio Access Network Application Part (RANAP) is one of the new additions to the SS7 protocol family, standardized for UMTS networks. User Parts and Application Parts do not change when the Message Transfer Part is replaced by SIGTRAN.

2.2 Message Transfer Part

The SS7 Message Transfer Part provides a connectionless, point-to-point transfer service used by protocols located in the upper layers of the SS7 architecture. Each signalling message contains two addresses called Origin Point Code and Destination Point Code. The MTP may send the message via a direct link from origin to destination or, most often, via one ore more transfer points that route the message towards its destination. The SS7 data-link layer protocol (MTP2) handles error control and retransmissions in each hop. When the signalling message reaches its destination, the Octet Indicator Service field indicates the upper-layer protocol to which the message has to be delivered. (This is similar to the function of port numbers in TCP and UDP.)

As mentioned above, signalling links and routes are normally provisioned redundantly, and the MTP performs load sharing among the available links and routes. As a result, two signalling messages addressed to the same destination may follow different routes and be delivered out of order (recall that MTP provides a connectionless service). For example, in Fig. 3 the possible routes from A to B in absence of failures are A-STP1-STP2-B, A-STP1-STP4-B, A-STP3-STP2-B, and A-STP3-STP4-B. The links between transfer points of the same pair, i.e. STP1-STP3 and STP2-STP4, are used only in case of failures.

The MTP supports a mechanism that allows the sender to force the same route for a sequence of messages addressed to a given destination point, so they are delivered in order. It makes use of a field called Signalling Link Selector (SLS) that is included in every message. If several messages are sent from A to B with the same selector value, the MTP must choose the same route for all of them. For example, all ISUP messages related to the same call are sent with the same link selector, so that they are delivered in sequence and the signalling dialog for that call is not altered.

The network layer protocol (MTP3) includes management procedures that monitor the status of the SS7 network and divert traffic from a failed route to an alternative one if necessary [5]. Fig. 3 shows different cases of link or node failures and the alternative routes that can be followed in each case to reach node B. In Fig. 3a) the link STP1-STP2 has failed. Therefore, STP1 suspends the load sharing between STP2 and STP4, and temporarily routes all messages in transit for destinations B and C via STP4 regardless of their link selectors. If STP1-STP4 also failed, STP1 would divert traffic to STP3 as a last option. Fig. 3b) shows a similar case. If STP2 goes down, both STP1 and STP3 divert traffic to STP4. Finally, in Fig. 3c) the link STP2-B has failed, but STP2 can still deliver messages to B via STP4.

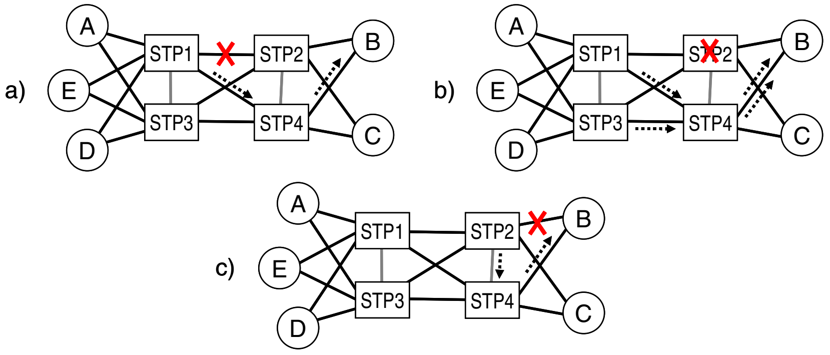


Fig. 3. Examples of Possible Failures and Alternative Routes

Network redundancy and the MTP routing functions outlined above offer a reliable transfer of messages between any two signalling points, meeting strict requirements, for example: message loss probability smaller than 10^{-7}, probability of message delivered out-of-sequence (including message duplication) smaller than 10^{-10}, probability of message delivered with undetected errors smaller than 10^{-10}, and signalling route set unavailability smaller than 10 minutes per year [6].

In addition, the MTP offers low message transfer delays (except for messages transmitted over satellite links, due to the long propagation delay). The delay in each intermediate transfer point is minimised by overdimensioning the capacity of the outgoing links. The performance and availability of SS7 networks based on mated STP pairs have been extensively studied in the literature [7].

3 Reliability in SIGTRAN Networks

The IETF SIGTRAN working group addressed the transport of signalling protocols (ISUP, TCAP, etc.) over IP, taking into account the functional and performance requirements of traditional signalling networks. For this purpose, the group has defined a general architecture [3] (see Fig. 1 and Fig. 4), the Stream Control Transmission Protocol (SCTP), and several adaptation protocols. The MTP3-User Adaptation Layer (M3UA) is the most important for this paper. SIGTRAN uses IP and existing IETF quality of service and security standards. As mentioned in the introduction, SIGTRAN is useful for new operators with IP networks that need to interwork with SS7 networks through signalling gateways, as well as for incumbents that are migrating their legacy circuit-switched infrastructure to IP. The functions of SCTP and M3UA that are more relevant for this paper are analyzed in sections 3.1 and 3.2 respectively.

3.1 Stream Control Transmission Protocol (SCTP)

SCTP [4,8] is a reliable, connection-oriented protocol that offers acknowledged, error-free, non-duplicated transfer of user messages. A connection established between two SCTP endpoints, called an *association*, may contain one or several *streams* in each direction. Sequence numbers are defined per stream, so SCTP guarantees ordered delivery within each stream only. In this way, message losses and retransmissions in one stream do not delay other messages going in the same direction but in a different stream. The stream identifier is 16 bits long, so there is ample room to define as many streams as necessary. Some SCTP messages may be marked with the U (unordered) bit set to 1. They are delivered to the upper layer as soon as they are received. The concept of stream in SCTP can be related to that of Signalling Link Selector. As stated in section 2.2, SS7 messages transmitted with different selector values are not required to maintain their relative order. Therefore, these messages may be transported over different SCTP streams.

SCTP supports multi-homing. Each association endpoint can have several IP addresses, and the protocol includes procedures for actively monitoring the reachability status of each peer address by means of periodic "heartbeat" messages. Each endpoint keeps a transmission error count for each of the addresses of its peer. An address is marked inactive if its error count exceeds a predefined threshold. When a heartbeat acknowledgement is received from a peer address, it is considered active again.

The destination and source IP addresses that an endpoint normally puts into outbound packets are called the primary path. Alternative active addresses, if available, are used for retransmissions or when the primary path is interrupted due to a network failure. If the underlying IP network topology is configured in such a way that packets sent to different peer addresses follow disjoint paths in the network without common points of failure, SCTP is able to maintain the association and provides effective recovery from network failures. If, in spite of multi-homing, the peer endpoint becomes unreachable, e.g. due to multiple network failures or to a failure of the endpoint itself, the SCTP association is lost. In this case, the redundancy mechanisms of the upper layer (see next section) come into play.

3.2 MTP3-User Adaptation Layer (M3UA)

In the SS7 architecture, ISUP and SCCP utilize the services provided by the MTP3 layer. The MTP3-User Adaptation Layer (M3UA) [9] has been designed to transport these protocols over IP without change. The 3rd Generation Partnership Project (3GPP) specifies the use of M3UA over SCTP as an option for signalling transport in the UMTS Terrestrial Access Network (UTRAN) interfaces Iub, Iur, and Iu.

M3UA can be used between a Signalling Gateway (SG) and an IP node, for example a Media Gateway Controller (MGC), an IP Home Location Register (HLR), etc. See Fig. 4. The M3UA gateway terminates all MTP protocols on the SS7 side.

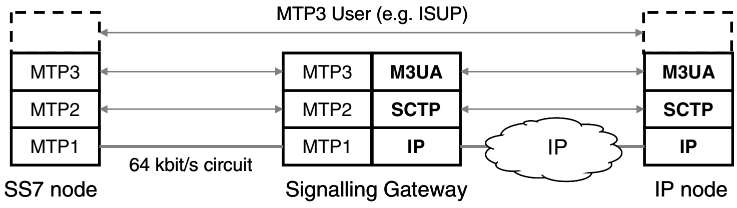


Fig. 4. Signalling Gateway with M3UA

From the point of view of the SS7 network, a link towards an M3UA gateway can route messages addressed to one or several signalling points, which terminate the higher-layer SS7 protocols (ISUP, SCCP, TCAP, MAP, etc.) Consequently, an M3UA gateway behaves as a Signalling Transfer Point (STP) in the SS7 terminology.

M3UA allows a completely distributed implementation of the protocol entities that process the SS7 messages on the IP side. The signalling messages that arrive from the SS7 network may be distributed by the gateway to physically different hosts located anywhere in the IP network, in order to achieve load sharing or redundancy. M3UA defines several logical entities intended to provide a flexible and dynamic allocation of signalling messages to protocol entities able to handle them. A Routing Key (RK) defines a subset of signalling messages based on conditions set upon the values of selected message fields. In a general example, RK1 may correspond to all messages addressed to destination X, and RK2 to ISUP messages addressed to destination Y.

An Application Server Process (ASP) is a process instance capable of handling messages for one or several RKs. At a given point in time, a process may be in different states for each of its RKs. For example, ASP A may be active for RK1 and RK2 (i.e. A is currently processing messages of those RKs), but inactive for RK3 (i.e. A does not process messages of RK3 now, but can do it later if necessary.) Each ASP is connected to the gateway via an SCTP association which transports all signalling messages handled by that process.

More than one ASP may be able to process messages for the same RK. For example, let us assume that processes A and B (possibly located in different hosts) can handle messages identified by RK1. One of them may be active and the other inactive as a backup. Alternatively, both processes may be active at the same time in load sharing (some messages go to A, and the others to B) or broadcast mode (each message goes to both A and B). The set of all Application Server Processes associated to a given routing key is called the Application Server (AS) for that key. See Fig. 5.

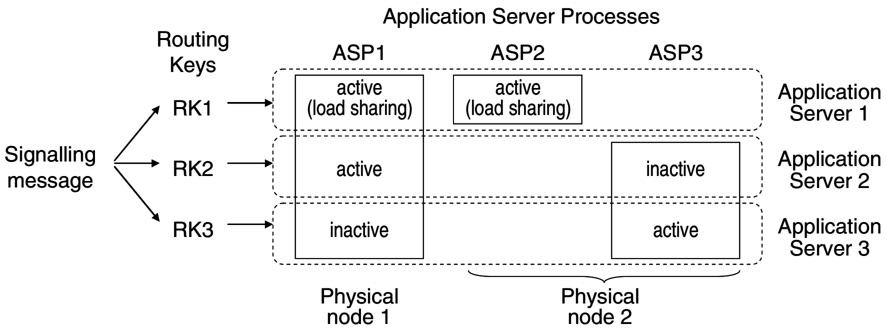


Fig. 5. Routing Keys, Application Server Processes, and Application Servers

The signalling gateway maintains a list of currently defined routing keys and monitors the state of all configured remote processes. For each signalling message coming from the SS7 network, the gateway matches the message to one key, determines the destination process (or processes in case of broadcast), and forwards it over SCTP.

In summary, by using M3UA with a redundant configuration of active and backup ASPs located in different machines, the SIGTRAN network can be effectively protected against host failures. A redundant IP network topology, together with the heartbeat and multi-homing procedures of SCTP associations presented in section 3.2, add protection against network failures. With an adequate configuration of M3UA and SCTP protocols, SIGTRAN networks can match the high reliability levels of traditional SS7 networks. The next section illustrates this with a detailed example.

3.3 Proposed SIGTRAN Network Configuration

Based on the analysis of relevant SIGTRAN mechanisms made above, this section defines a SIGTRAN network design equivalent to the basic mesh or quad of Signalling Transfer Points that typically forms the building block of SS7 networks. (This basic mesh, explained in section 2, is reproduced in Fig. 6. For clarity of the explanation, no links between transfer points of the same pair, i.e. STP1-STP3 and STP2-STP4, have been included in the example.) This provides a general solution in which large SIGTRAN networks are built by systematically repeating the configuration of interconnected signalling gateway pairs described here.

Let us recall that Signalling Transfer Points (STPs) are deployed in pairs, and each STP pair gives access to a subset of Signalling Points (SPs). In absence of failures, the traffic of each area is shared by the two STPs serving that area. For example, in Fig. 6 STP1 and STP3 serve area L, while STP2 and STP4 serve area R. STP1 will normally split the traffic addressed to destinations located in area R among STP2 and STP4 depending on the Signalling Link Selector (SLS) values. Let us assume that messages with even selector values are sent via STP2, and messages with odd values are sent via STP4. If the link STP1-STP2 fails, or if STP2 is down, STP1 will route all traffic for area R through STP4. Conversely, if the failure affects STP4, STP1 will route all traffic through STP2. The other three transfer points behave in a similar way.

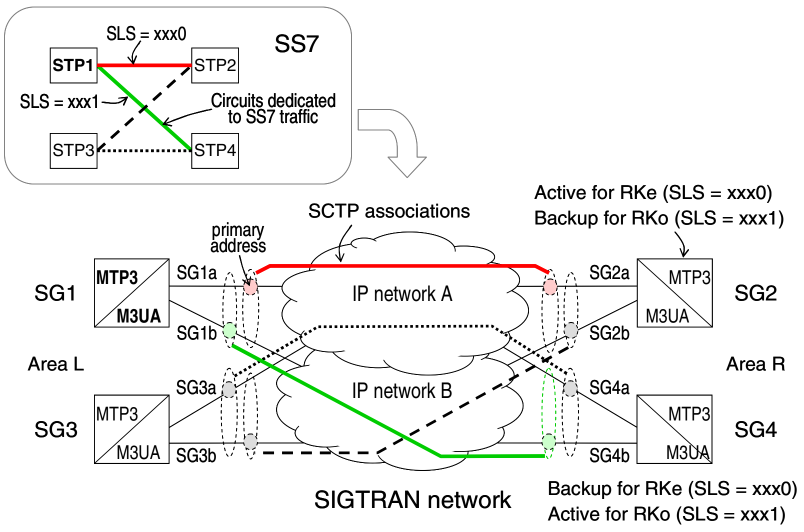


Fig. 6. Basic SS7 Mesh Network and Equivalent SIGTRAN Network

This behaviour can be reproduced in a SIGTRAN network with M3UA as follows. Each transfer point is upgraded to a Signalling Gateway (SG) that implements the MTP protocol stack on the links that come from the end points in its service area, and the M3UA/SCTP/IP stack on the links that go to gateways serving other areas. Each signalling gateway is assumed to have separate interfaces to two IP networks A and B. Let SG1a and SG1b denote the IP addresses of the two network interfaces in gateway SG1. (In the following we take SG1 as example. The other three gateways are configured in a similar way.)

The signalling link SG1–SG2 is implemented as a multi-homed SCTP association between the address list (SG1a*, SG1b) on one side and (SG2a*, SG2b) on the other side. A second SCTP association is established between (SG1a, SG1b*) and (SG4a, SG4b*), corresponding to the link SG1–SG4. The stars indicate the addresses selected for the primary path in each association. If SCTP detects that the primary path is unavailable, e.g. due to a network failure, it automatically switches traffic to the alternative path without breaking the association. Therefore, from the point of view of M3UA the network failure is transparent and the connectivity between the signalling gateways is maintained.

As indicated above, gateway SG1 must send messages with even link selector values to STP2, and messages with odd values to STP4. To force this behaviour, we define two routing keys: RKe matches traffic going to SG2/SG4 with even selector values and RKo matches traffic with odd values. RKe is associated with an Application Server formed by two Application Server Processes: SG2 (active) and SG4 (backup). RKo is associated with another server formed by the same two processes as before, but now the initial active and backup roles are reversed: SG4 is the active process and SG1 is the backup. In this way, if SG2 fails (or the SCTP layer is unable to maintain the association SG1–SG2) SG1 will send all traffic to SG4 as expected. Obviously, if the failure affects SG4, traffic goes to SG2.

Finally, in SS7 networks the MTP is not required to maintain the relative order of messages with different Signalling Link Selector values. Therefore, in the proposed SIGTRAN network different selector values may be mapped to different streams within each SCTP association between gateways.

Based on the preceding description, it is straightforward to deduce the configuration of SCTP associations and active/backup processes for the remaining gateways SG2, SG3, and SG4. It is also possible to generalize this example to include links between STPs of the same pair or for networks with more than two STP pairs.

The proposed configuration can be applied, for example, by circuit-switched network operators that have deployed large SS7 networks (fixed-line incumbents, mobile operators) and want to gradually migrate to IP and SIGTRAN. In this case, an initial step may be moving the inter-STP traffic to IP. In addition to the redundancy and failure recovery capabilities provided by M3UA and SCTP, the existing signalling links can be temporarily kept as additional alternative routes until the reliability of the new SIGTRAN network has been validated. In a second step, the access links between Signalling Points and STPs may be migrated to SIGTRAN as well.

4 Evaluation of Signalling Load in the SIGTRAN Network

The SIGTRAN traffic will increase the load in the IP network of the operator. An approximate estimation of the SIGTRAN traffic volume can be obtained from measurements taken in the existing SS7 network, adjusting the values to take into account the different protocol overheads of MTP and SIGTRAN. While MTP2 and MTP3 add a total of 11 octets per message, the combined headers of M3UA, SCTP, and IP are considerably longer. M3UA data messages include a common header (8 octets), an additional header (16 octets), and two optional fields (8 octets each), giving a total of 40 octets. An SCTP message with common header (12 octets), DATA block and Selective Acknowledgement block (16 octets each) adds 44 octets more. Adding an IP v4 header (20 octets), the resulting overhead is 104 octets per message, plus the overhead of any protocol layers below IP. This value may be further increased if IPsec security procedures are used [10]. Considering that many of the messages generated by ISUP and other SS7 protocols are short, e.g. a few tens of octets, it is clear that the SIGTRAN overhead has a significant impact on the total signalling traffic load.

The estimation of the SIGTRAN traffic volume is useful if this traffic is going to receive a preferential treatment in the IP network, for example by means of Differentiated Services (diffserv), MPLS Traffic Engineering (MPLS-TE), or a combination of both [11]. These mechanisms help to offer a better quality of service (e.g. low delay) to signalling flows transported over the IP network. Additionally, MPLS offers fast reroute capabilities below IP [12] that reduce the number of network failures that have to be recovered by SCTP or M3UA.

Besides reliability, quality of service and security are two fundamental requirements for every signalling network, including IP-based ones. Although IPsec, diffserv, and MPLS-TE are not considered further in this paper, they are standards that complement SIGTRAN in order to meet such requirements. Therefore, the necessity of using them should be assessed as part of the overall SIGTRAN network design.

Instead of relying on measurements taken in a real SS7 network, or as a complement to them, the signalling traffic may be modelled analytically or by simulation. The effort required to develop a detailed signalling model is not small, because it is necessary first to identify the services and procedures of interest, and then to analyze the possible SS7 message exchanges generated by each of them. However, the advantage of such model is that it relates signalling load to service usage. Consequently, it serves to study the signalling network not only in the current situation, but also in alternative scenarios that may be of interest for the operator and for which there are no measurements. For example, the model can be used to estimate the impact on the signalling network caused by changes in current services or user profiles, introduction of new services, etc.

To illustrate this, we consider here the signalling messages used in GSM mobile networks to support voice calls and the Short Message Service (SMS), including the signalling procedures triggered by user mobility, for example change of Visitor Location Register. For each service, we have analyzed the possible cases that produce different signalling sequences. For example, in voice calls it is necessary to distinguish calls coming from an external network, calls going to an external network, intra-network calls between mobile users in the same region, calls between mobiles in different regions, etc. For each service and case, we identify the network nodes involved: access and transit Mobile Switching Center (MSC), Home Location Register (HLR), Visitor Location Register (VLR), Short Message Service Gateway (SMSG), Serving GPRS Support Node (SGSN), Signalling Transfer Point (STP), etc., as well as the signalling messages transmitted and received by each of them.

For example, Table 1 shows the messages exchanged during a call between two mobile users located in the same region (so their respective switching centers are connected to the same pair of STPs). Similar tables for other services are not shown here due to lack of space, but they can be found in [13].

Table 1. Signalling Messages During a Voice Call

Network node		**SS7 Integrated Services User Part (ISUP) Messages**	**SS7 Mobile Application Part (MAP) Messages**
Calling MSC	Send	IAM, REL	SRI
	Receive	ACM, ANM, RLC	SRI_Ack
Called MSC/VLR	Send	ACM, ANM, RLC	Provide_MSRN_Ack
	Receive	IAM, REL	Provide_MSRN
Transit MSC	Send	IAM, ACM, ANM, REL, RLC	
	Receive	IAM, ACM, ANM, REL, RLC	
HLR	Send		SRI_Ack, Provide_MSRN
	Receive		SRI, Provide_MSRN_Ack

The analysis includes several parameters that may be configured in order to adapt it to the scenarios of interest, namely network topology, number of users, traffic per user and service in the busy hour, message lengths, and protocol overheads. As result, we obtain the signalling load in each node interface, and the load in the links between STPs. For example, let us assume a GSM network with 10 million subscribers uniformly distributed over 50 MSCs and 10 HLRs. The network has 8 transit MSCs, 8 SGSNs, and 4 SMSGs. The SS7 network consists of three pairs of STPs interconnected by a full mesh of signalling links. The parameter values that define the traffic per subscriber have been derived from real data of the Spanish mobile market, taken from the annual report published by the Spanish national telecommunications regulator 'Comisión del Mercado de las Telecomunicaciones' (CMT) [14]. Based on this report, we have selected values for short messages sent and received per subscriber, voice calls with subscribers of the same mobile network, calls with subscribers of other mobile networks, and calls with subscribers of fixed telephone networks.

If SIGTRAN is deployed in all interfaces (i.e. inter-STP links and access links of MSCs, HLRs, etc.), the estimated signalling load values computed by the model are shown in Fig. 7. The highest load, 1482 kbit/s, is found in the transit MSCs. MSC/VLRs, HLRs, and SMSGs give smaller values, between 146 and 355 kbit/s. For each node, the graph indicates the fractions of the total load generated by the different services considered.

Note that only MSC/VLRs and HLRs are involved in all services. Transit MSCs are not concerned by SMS or location updates, and SMSGs deal only with the Short Message Service. The small signalling load computed for SGSNs, 4.3 kbit/s, corresponds to routing area updates where subscribers move from their current node to a new one.

The values given in Fig. 7 correspond to the flow of messages generated by each network node. The signalling load in the opposite direction, i.e. messages received by each node, may be higher or lower because the procedures are not symmetric. Although not shown in the graph, we estimated the load values in both directions and the differences found are small: between 0.6% and 5.3%.

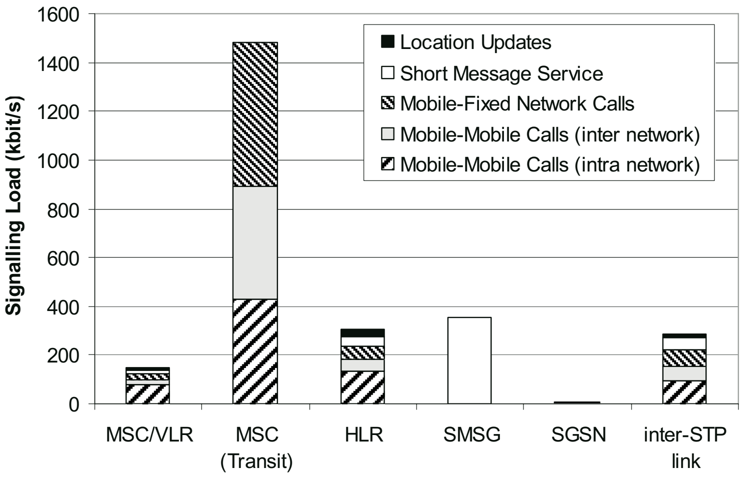


Fig. 7. Signalling Load when All Nodes Use SIGTRAN

The type of data commented above helps to quantify the new signalling traffic flows that must be transported over IP if the traditional SS7 network is migrated to SIGTRAN, in order to ensure that these flows, which are crucial for the operator, receive an adequate quality of service.

5 Related Work

As mentioned previously, multi-homing and other SCTP features have been extensively studied, but there are comparatively few references in the literature about the M3UA protocol. The operation of SCTP in failover scenarios has been investigated for example in [15,16,17]. These papers evaluate SCTP-controlled failovers under different parameter settings (e.g. timeout values, round-trip delay, traffic load, etc.) The correctness of the failover algorithm has been analysed in [18], where the authors have detected problems in some circumstances and proposed solutions. Other works study applications of SCTP beyond its original purpose of signalling transport, such as HTTP transport over SCTP [19] and mobility management at the transport layer based on multi-homing plus extensions for dynamic address reconfiguration [20].

The configuration of process redundancy mechanisms in M3UA is touched upon in [21] and [22], but we have not found any papers dealing with this issue in detail. In [23], the authors point out some limitations of redundancy procedures in M3UA and other SIGTRAN adaptation protocols, and present solutions based on the new Reliable Server Pooling architecture being defined by the IETF. Rserpool aims at creating a unified redundancy solution supporting high availability and scalability of applications through the use of pools of servers. At the time of writing this paper, only the requirements for reliable server pooling have been published as an informational RFC, while the architecture and the protocols are still in draft state.

6 Conclusions

This paper focuses on the practical application of the IETF SIGTRAN standards in order to facilitate the migration towards IP of traditional SS7 networks used by telecommunications operators. Special attention is paid to the high network availability and quality of service required by signalling traffic when it is moved from SS7 to IP.

The paper makes two contributions. Firstly, it describes a configuration of SIGTRAN gateways with SCTP multi-homed associations and M3UA redundant servers that matches the signalling route redundancy offered by the basic arrangement of meshed STP pairs typically employed in SS7 networks. This configuration provides a general building block that can be used to deploy large SIGTRAN networks. Secondly, it presents a detailed analysis of signalling message exchanges that serves to estimate the SIGTRAN network load in different network configuration and service scenarios. As a practical case, we describe the application of the proposed SIGTRAN configuration to a GSM mobile network, and evaluate the resulting signalling load, taking as input real data of subscriber behaviour in Spanish mobile networks.

Acknowledgements

The authors acknowledge the initial support of ‘Cátedra Orange’ in Universidad Politécnica de Madrid, and partial funding by the Spanish Ministry of Education and Science under the project CASERTEL-NGN (TSI2005-07306-C02-01).

References

1. TU-T Recommendation Q.700: Introduction to Signalling System No. 7 (1993)
2. Kuhn, P.J., Pack, C.D., Skoog, R.A.: Common channel signalling networks: past, present, future. IEEE Journal on Selected Areas in Communications, vol. 12, no. 3 (1994)
3. Ong, L., Rytina, I., García, M., Schwarzbauer, H., Coene, L., Lin, H., Juhasz, I., Holdrege, M., Sharp, C.: Framework Architecture for Signalling Transport. RFC 2719 (1999)
4. Stewart, R.: Stream Control Transmission Protocol. RFC 4960 (2007)
5. ITU-T Recommendation Q.704: Signalling network functions and messages (1996)
6. ITU-T Rec. Q.706: Message Transfer Part signalling performance (1993)
7. Chung, M.Y., You, J.U., Sung, D.K., Choi, B.D.: Performability Analysis of Common-Channel Signalling Networks Based on Signalling System #7. IEEE Transactions on Reliability, vol. 48, no. 3 (1999)
8. Fu, S., Atiquzzaman, M.: SCTP: state of the art in research, products, and technical challenges. IEEE 18th Annual Workshop on Computer Communications (CCW), Dana Point, California, USA (2003)
9. Morneault, K. (ed.), Pastor-Balbas, J. (ed.): Signalling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA). RFC 4666 (2006)
10. Loughney, J., Tüxen, M., Pastor-Balbas, J.: Security Considerations for Signalling Transport (SIGTRAN) Protocols. RFC 3788 (2004)
11. Fineberg, V.: QoS Support in MPLS Networks. MPLS/FR Alliance (2003)
12. Pan, P. (ed.), Swallow, G. (ed.), Atlas, A. (ed.): Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090 (2005)
13. Guénon, G., Vázquez, E., Álvarez-Campana, M.: SS7 Signalling Transport over IP. Universidad Politécnica de Madrid (2006)
14. Comisión del Mercado de las Telecomunicaciones (CMT), http://www.cmt.es
15. Jungmaier, A., Rathgeb, E., Tüxen, M.: On the use of SCTP in failover scenarios. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI), Orlando, Florida, USA (2002)
16. Grinnemo, K-J., Brunstrom, A.: Impact of Traffic Load on SCTP Failovers in SIGTRAN. 4th International Conference on Networking (ICN), Reunion Island (2005)
17. Jungmaier, A., Rathgeb, E.: On SCTP multi-homing performance. Telecommunication Systems, vol. 31, no. 2-3 (2006)
18. Noonan, J., Murphy, J., Murphy, S., Perry, P.: Stall and Path Monitoring Issues in SCTP. IEEE INFOCOM 2006, Barcelona, Spain (2006)
19. Natarajan, P., Iyengar, J.R., Amer, P.D., Stewart, R.: SCTP: an innovative transport layer protocol for the web. 15th International Conference on World Wide Web, Edinburgh, Scotland (2006)
20. Koh, S.J., Chang, M.J., Lee, M.: mSCTP for soft handover in transport layer. IEEE Communications Letters, vol. 8, no. 3 (2004)
21. Gradischnig, K.D., Tüxen, M.: Signalling transport over IP-based networks using IETF standards. Third International Workshop on the Design of Reliable Communication Networks, DRCN'01, Budapest, Hungary (2001)
22. Lee, H., Lee, B.: A redundancy method of AS traffic in signalling gateway using load sharing scheme. 6th International Conference on Advanced Communication Technology, Phoenix Park, Korea (2004)
23. Dreibholz, T., Rathgeb, E.P.: RSerPool - Providing Highly Available Services using Unreliable Servers. IEEE EuroMicro 2005, Porto, Portugal (2005)

An Experimental Investigation of the End-to-End QoS of the Apple Darwin Streaming Server

Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano

Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari,
Via Re David 200, Italy
{ldecicco,mascolo,vpalmisano}@poliba.it

Abstract. Video content distribution over the traditional best-effort, store-and-forward Internet Protocol is of ever increasing importance due to the great success of new web services such as personal video broadcast or television over IP (IPTV). In this paper we investigate the end-to-end quality of service (QoS) that is provided by the Apple Darwin Streaming Server and the Quick-Time client player in the presence of time-varying available bandwidth and multiple concurrent streaming sessions. The considered end-to-end QoS parameters are the loss rates and the friendliness experienced when the available bandwidth changes and when multiple QuickTime streaming sessions and/or TCP sessions compete in order to obtain a bandwidth share.

We found that the Darwin Streaming Server implements a TCP-like congestion control that is more aggressive than TCP; in particular, when more QuickTime flows share the same link with TCP flows, QuickTime gets more bandwidth than TCP. Moreover, when more QuickTime flows share the same link, they exhibit a high loss rate.

Keywords: End-to-End QoS, Multimedia Congestion Control, Reliable UDP, Apple Darwin Streaming Server, QuickTime Player.

1 Introduction

Audio/Video content distribution is nowadays a potential killer application for the Internet as it is proved by the great success of YouTube[1] and by the introduction of new applications such as Joost[2] and Babelgum[3], which aim at providing television distribution over IP. The most part of Internet traffic is still delivered using the TCP transport protocol, which has been the key factor of Internet stability so far. This is the reason for which many Web sites (such as YouTube) that host small length and low resolution videos use only pseudo-streaming technologies that are based on the simple TCP download. In this way the generated traffic is not harmful for the stability of the Internet because the TCP transport protocol implements an effective congestion control algorithm [1]. However, it is

[1] http://www.youtube.com/
[2] http://www.joost.com/
[3] http://www.babelgum.com/

not clear if the perceived quality is satisfactory for the user. In fact the source of the great success obtained by YouTube is very much likely to be due to the richness of contents and its large user base rather than to the quality of the video delivering.

The TCP window-based congestion control guarantees congestion avoidance by using the additive increase/multiplicative decrease paradigm [1] and reliable delivery of the content through packet retransmissions but not content delivery within delay constraints. On the other hand, multimedia streaming services can tolerate some low packet loss percentage but require more tight quality of service (QoS) requirements in terms of end-to-end delays and jitter. For this reason the UDP protocol is the preferred transport protocol for multimedia streams, because, as matter of fact, it is a simple packet multiplexer/demultiplexer, where the packet sending rate can be managed at the application level. However, many multimedia applications which use UDP do not implement effective congestion control mechanisms, thus possibly leading to a network *congestion collapse* [2] due to the presence of unresponsive flows on the same bottleneck link. This circumstance can cause a high loss rate, which is an important factor that affects the perceived quality [3,4].

Several efforts have been made to design multimedia congestion control protocols that are TCP friendly, where friendliness here means that the multimedia flows will share the network bandwidth with TCP flows fairly. The TCP Friendly Rate Control (TFRC) [5] protocol and the Datagram Congestion Control Protocol (DCCP) framework [6] are two IETF standards proposed as possible congestion control algorithms for the transport of multimedia flows. An interesting solution is the Reliable UDP proposed by Apple, which is a TCP-like congestion control protocol that aims at providing a set of QoS enhancements for RTP multimedia flows [7] (see Sec. 3 for more details).

The Darwin Streaming Server (DSS) is the open source version of the commercial Apple's QuickTime Streaming Server (QTSS) that allows the distribution of streamed multimedia contents over the Internet. The protocols employed by DSS are the standard RTP and RTCP[4]. DSS is based on the same code base of QTSS, but its source code is freely distributed under the Apple Public Source License. Both DSS and the official commercial QuickTime Player (QTP)[5] implement the Reliable RTP congestion control. DSS uses well-known standards (such as RTP, RTCP, SDP and HTTP) for content distribution. Thus, every multimedia player that supports RTP can be used as client.

In this paper we have used the official DSS and QTP for investigating the effectiveness of Reliable UDP congestion control algorithm in the presence of changing available bandwidth and/or packet losses. The goal of these investigations is to evaluate how the congestion control algorithm implemented by Reliable UDP allows the sending rates be managed in order to match the available bandwidth when multiple streaming sessions and/or TCP connections share the same link, thus revealing intra-protocol and inter-protocol fairness behaviour.

[4] http://developer.apple.com/opensource/server/streaming
[5] http://www.apple.com/it/quicktime/download/

The rest of the paper is organized as follows: Section 2 presents the previous work on streaming server performance evaluations; Section 4 describes the considered experimental testbed and the scenarios; Section 5 reports the experimental results and finally Section 6 draws the conclusions.

2 Related Work

Many recent investigations have focused on multimedia streaming client/server applications. In [8] an investigation on the Internet streaming quality and efficiency is performed by collecting connection data from thousands of broadband home users accessing both on-demand and live streaming media. Authors have found that input rate adaptation, even though implemented in media authoring, is poorly utilized, particularly when a pre-buffering phase is used. The pre-buffering phase (also called *Fast Streaming*) is widely used and much quality degradation is caused by re-buffering events.

Authors of [9] present an evaluation of RealVideo streaming over UDP and over TCP. They have found that RealVideo over UDP does not respond to Internet congestion by adapting the sending and/or the encoding rate. In particular, under very constrained bandwidth conditions RealVideo UDP streams do not share the bandwidth fairly with concurrent TCP connections. Moreover, authors report that only the 35% of RealServer implement some form of encoding scalability (called *Media Scaling*), and less then the 50% of the clips were using more then 4 encoding levels so that they can only adapt to the available bandwidth coarsely.

In [10] an investigation of the Windows Streaming Media (WSM) is performed in order to analize content multiple encoding. They found that, if the network capacity is lower than the minimum available encoding level, WSM produces high packet loss rates and it is unfair with concurrent TCP flows.

In [11] a comparative analysis of RealPlayer, Windows Media Player and Quicktime is performed. They used a UDP cross traffic generator concurrent with each multimedia stream flow in order to emulate a network congestion condition. Authors have found that Quicktime provided the lowest packet loss rate among the applications, thus indicating that DSS performs an effective congestion control algorithm as compared to the other media streaming solutions.

In this work we aim at performing an extensive investigation of DSS by testing it on different scenarios in order to:

1. evaluate how Reliable UDP reacts to congestion episodes;
2. estimate the friendliness between more concurrent QuickTime multimedia flows and between QuickTime flows and TCP flows.

3 Apple's Reliable UDP

Reliable UDP is a set of extensions to the RTP protocol designed in order to provide retransmission and congestion control mechanism to the unreliable UDP protocol. These extensions allow multimedia streams to behave like TCP flows, while providing soft real-time features. Apple's version of Reliable UDP implements a congestion control based on the Additive Increase/Multiplicative Decrease approach: the sender maintains a *congestion window* (CWND)such as the one used by the TCP congestion control; during the *slow-start* phase, for each ACKed packet, CWND increases by 1 segment, whereas during the *congestion avoidance* phase CWND increases by 1 segment every *round trip time* (RTT). When a timeout expires (which in the DSS implementation is always equal to 250 ms) or 3 duplicate ACKs (3DUPACKs) are received the slow-start threshold is set to 3/4 of CWND and CWND is halved. This behaviour makes Reliable UDP more aggressive than TCP when a loss event is detected, because when 3DUPACKs are received TCP halves the slow-start threshold and enters the congestion avoidance phase, whereas Reliable UDP enters the slow-start phase and then the congestion avoidance phase.

The tests we have carried on confirm this behaviour and show that Reliable UDP tends to use more bandwidth with respect to TCP concurrent flows.

4 Experimental Testbed

The testbed we have set up is made of a Linux machine, hosting the DSS, and a Windows machine where the QuickTime players have been installed. The Linux machine has beed equipped with a tool developed by us (`ipqshaper)` that performs bandwidth and delays shaping, and introduces packets losses. This tool uses the Application Programming Interface (API) provided by `Netfilter` [12] in order to redirect the packets sent or generated by the applications to a user-space drop-tail queue, where traffic shaping and measurement are performed.

As it is shown in Figure 1, the packets generated by DSS are redirected to `ipqshaper`'s queue, where traffic shaping is performed. The outbound packets from DSS are sent to the QuickTime players (P_1,..., P_n) that are installed on the Windows machine. `iperf` (T_1,..., T_n) were installed in order to generate TCP concurrent flows. The queue size was set equal to $10\,KB$, and a $10\,ms$ delay was applied to all outgoing packets.

It should be noticed that the experimental testbed is strictly equivalent to the one that could be obtained using a Dummynet-like router [13], the only difference being that in our case we are able to use only two hosts instead of three.

The video flows are generated by using the benchmark video sequence *Foreman*, which is encoded into an MPEG4 format, at a resulting average bitrate B_{avg} of $242\,kb/s$. We used the commercial version of QuickTime Player in order to produce the correct *hinted* .mov files, required by the DSS for the streaming.

The experimental scenarios we have tested are made of:

1. Multiple QuickTime concurrent flows with a super imposed constant bandwidth limitation;
2. Multiple QuickTime concurrent flows with a variable bandwidth limitation;
3. Single and Multiple Quicktime flows with a super imposed loss rate;

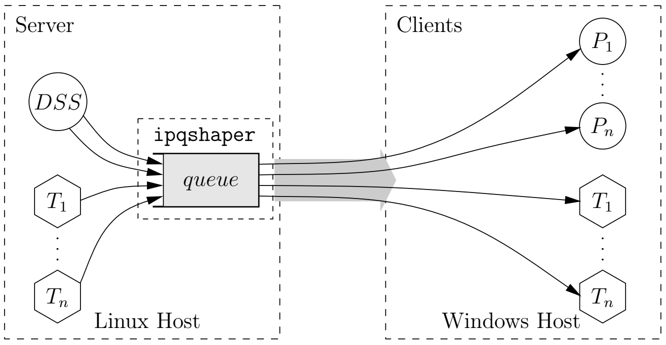


Fig. 1. Experimental testbed

4. Multiple QuickTime and TCP concurrent flows with a constant bandwidth limitation;
5. Multiple QuickTime and TCP concurrent flows with a variable bandwidth limitation.

The scenario (3) has been realised using the Gilbert model in order to emulate loss events that affect noisy channels (e.g. IEEE 802.11a/b/g connections [14]). This model uses a two-states Markov chain: a *good* and a *bad* state. When the process is into the *good* state, no loss events are generated. When in the *bad* state, the arriving packets are dropped with probability 1. If the transition probabilities are p_{good} and p_{bad}, it can be proved that the expected values are: $l_{good} = \frac{1}{1-p_{good}}$ and $l_{bad} = \frac{1}{1-p_{bad}}$. The values $l_{good} = 11.63$ and $l_{bad} = 1.78$ used here are the ones reported in [14] for IEEE 802.11 connections.

Throughput $r(t)$, loss rate $l(t)$ and goodput $g(t)$ shown in test results are defined as follows:

$$r(t) = \frac{S(t) - S(t - \Delta T)}{\Delta T},\ l(t) = \frac{L(t) - L(t - \Delta T)}{\Delta T},\ g(t) = r(t) - l(t)$$

where $S(t)$ and $L(t)$ are the number of sent bits and lost bits at time t respectively. We have considered $\Delta T = 0.5\,\mathrm{s}$ in our measurements.

We evaluate the link utilization (LU) as follows:

$$LU = \frac{\sum_{i=1}^{N} R_i}{C} \cdot 100$$

where R_i is the average throughput of the i-th flow, N is the number of concurrent connections accessing the bottleneck and C is the link capacity.

In order to evaluate the fairness, we employ the Jain Fairness Index (JFI) [15]. Moreover, in order to evaluate the temporal evolution of the fairness index, we define the instantaneous JFI as follows:

$$JFI(t) = \frac{\left(\sum_{i=1}^{N} g_i(t)\right)^2}{N \cdot \sum_{i=1}^{N} g_i^2(t)}$$

Where $g_i(t)$ is the goodput of the i-th flow and N is the number of concurrent connections accessing the bottleneck.

5 Results

In this section we present the more significative experimental results we have obtained.

5.1 Darwin Streaming Server Evaluation

Two and Four QuickTime Concurrent Flows. In this scenario we tested DSS in presence of 2 or 4 concurrent streaming sessions over the same link with super-imposed bandwidth reductions in order to evaluate the fairness of Reliable UDP.

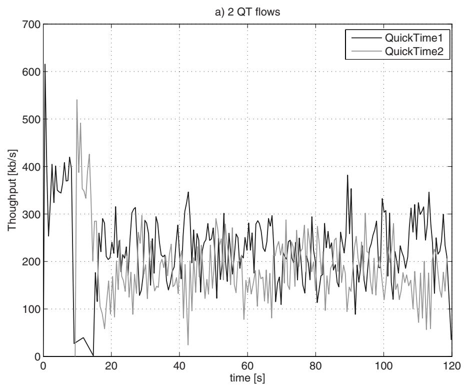
In the case of 2 concurrent QTP streaming sessions we imposed a bandwidth equal to $0.75 \cdot B_{avg} \cdot 2 = 480\, kb/s$, whereas in the case of 4 concurrent sessions we imposed a bandwidth equal to $0.75 \cdot B_{avg} \cdot 4 = 960\, kb/s$. Each connection starts after a $10s$ delay from the other in order to avoid overlaps between the pre-buffering phases.

As it is shown in Figure 2, during the first $10\, s$ of each connection, each flow tends to take up all the available bandwidth in order to fill the playout buffer (over-buffering phase). After this phase, DSS tries to allocate for each flow a bandwidth equal to the average streaming rate B_{avg}, but the first started flows tends to occupy more resources, as shown on Table 1. Moreover, the last started flows experience a higher loss rate, i.e. the adopted congestion control is aggressive and tries to reallocate bandwidth continuously. The link utilization in the case of two QT flows results equal to 79% whereas in the case of four QT flows results equal to 76%, that is, QT flows are unable to fully utilize the link capacity.

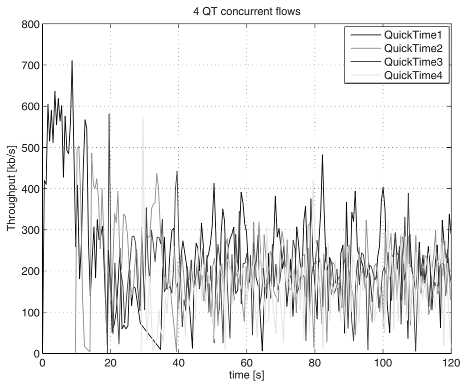
As it is shown in Figure 3, the JFIs oscillate in the range [0.5, 0.98] around their average values, meaning that the DSS congestion control becomes unfair with respect to other QT flows when the number of concurrent connections accessing the same bottleneck raises.

One and Four QuickTime Concurrent Flows Sharing a Square Wave Available Bandwidth. In this scenario we used an available bandwidth that varies as a square wave with maximum value $A_M = 1600 \cdot N$ kb/s and minimum value $A_m = 100 \cdot N$ kb/s (where $N = 2, 4$ is the number of concurrent flows) with period equal to $40\, s$ in order to investigate how DSS adapts his sending rate.

As it is shown in Figure 4(a), DSS adapts its sending rate in order to match the available bandwidth, using a pre-buffering phase every time a larger bandwidth is available. After each pre-buffering phase (that lasts roughly 10 s), the throughput is back to B_{avg}.

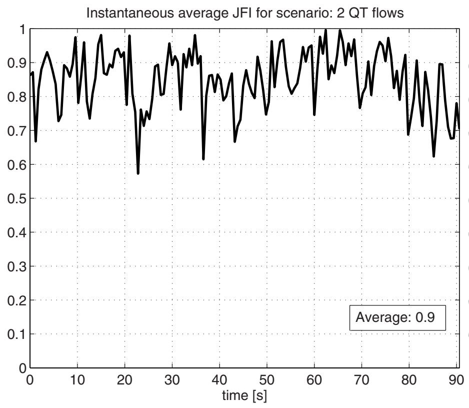


(a)

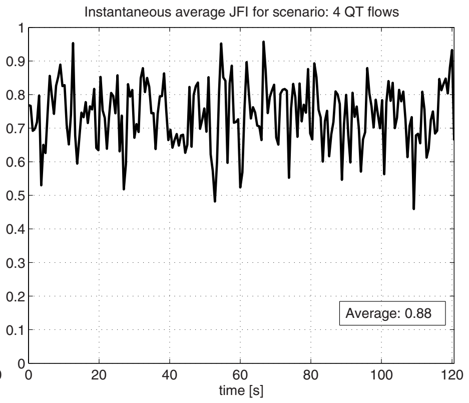


(b)

Fig. 2. Two and four QuickTime concurrent flows



(a)



(b)

Fig. 3. JFI index for two and four QuickTime concurrent flows

In the case of four QuickTime concurrent flows, we started all flows at the same time. As it is shown in Figure 4(b), when the available bandwidth increases from 100 to 1600 kb/s, each client starts an over-buffering phase. Therefore, in this case all clients equally compete for a bandwidth share. In fact Table 1 shows that the four clients loss rate levels are roughly the same and the JFI is close to 1 (fair share). We can infer that in this case DSS congestion control is more fair than the case described in the previous section, likely because the over-buffering phases start at the same time for each client.

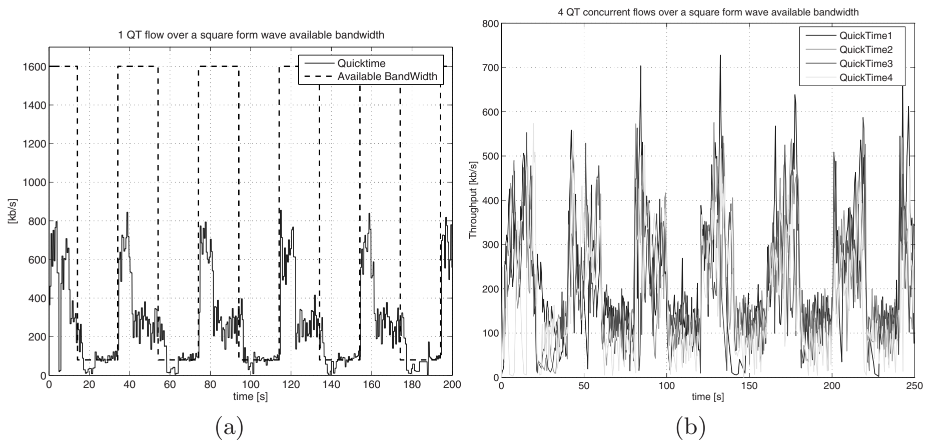


Fig. 4. One and four QuickTime concurrent flows over a square wave available bandwidth

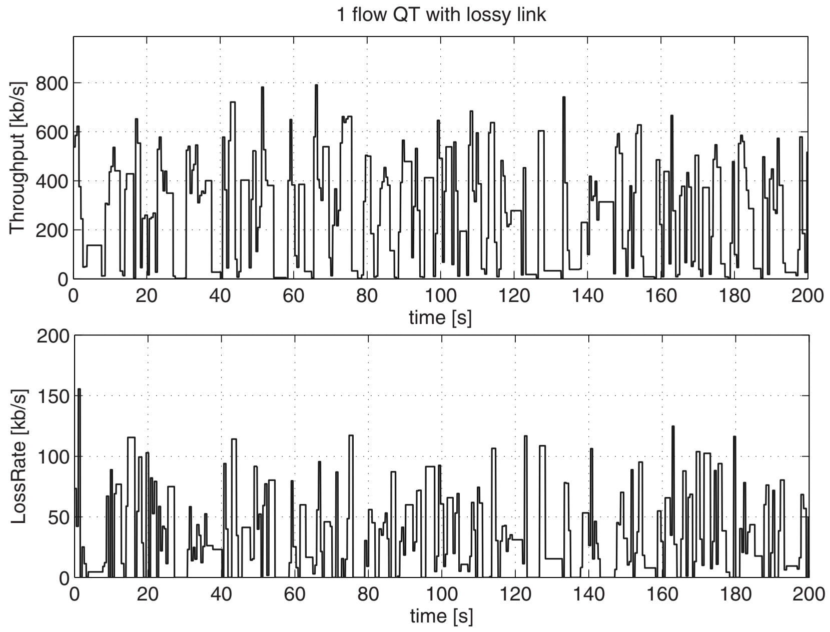


Fig. 5. One QuickTime flow over a lossy link

One QuickTime Flow over a Lossy Link. In this scenario we have evaluated the behaviour of DSS and QTP in the presence of a lossy link, emulated using a Gilbert model. Figure 5 shows the throughput and the loss rate imposed by

Table 1. DSS test results showing average throughputs (TP), lossrates (LR) and Jain Fairness Indexes (JFI)

Experiment	Flow	TP (kb/s)	LR (kb/s)	LR%	JFI
2 QT, constant bw	QT1	208.34	18.36	9	0.98
	QT2	171.97	26.83	16	
4 QT, constant bw	QT1	217.81	30.56	14	0.96
	QT2	183.37	37.15	20	
	QT3	178.14	48.21	27	
	QT4	154.34	51.54	33	
1 QT, variable bw	QT	257.67	10.91	4	-
4 QT, variable bw	QT1	212.52	24.96	12	0.99
	QT2	202.49	31.35	15	
	QT3	195.53	36.32	19	
	QT4	163.05	28.73	18	
1 QT, lossy link	QT	203.45	25.70	13	-

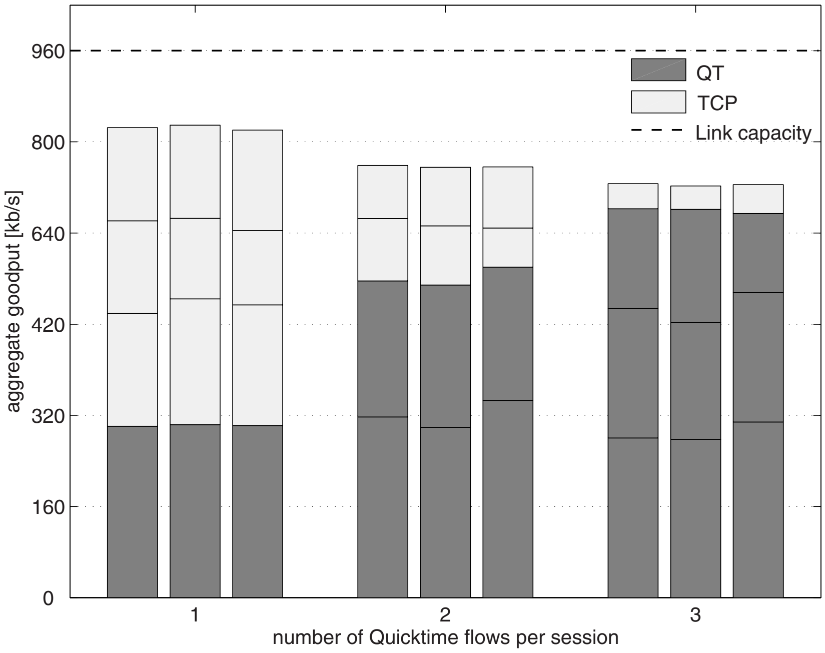


Fig. 6. Quicktime flows with concurrent TCP flows

the Gilbert model. As it is shown in Table 1, in this case the throughput is roughly equal to B_{avg}, i.e. the congestion control implemented by DSS exhibits a fast bandwidth reallocation, likely because Reliable UDP uses a slow-start phase after each loss event.

5.2 Darwin Streaming Server vs. TCP

Many QuickTime Flows with Many Concurrent TCP Flows. In this scenario we evaluated the TCP-friendliness of QuickTime streams using more concurrent connections: 1 QT and 3 TCP streams, 2 QT and 2 TCP streams and finally 3 QT and 1 TCP stream. We imposed an available bandwidth equal to $0.75 \cdot B_{avg} \cdot 4 = 960\,kb/s$. As it is shown from figure 6, QT flows tend to be unfair with respect to TCP flows. By looking at Table 2, in the case of 2 QT and 2 TCP flows, the calculated JFI is low (0.81), instead in the case of 3 QT and 1 TCP flows the JFI is 0.87.

The evaluated link utilization in the case of 1 QT and 3 TCP is equal to 81%, whereas in the other cases results equal to 73%, i.e. TCP flows are able to utilize more bandwidth than the QT flows.

Many QuickTime Flow with Many Concurrent TCP Flows over a Square Wave Available Bandwidth. In this scenario we have evaluated the TCP-friendliness of QuickTime streams in presence of time-varying available bandwidth, that varies as a square wave with maximum value $A_M = 1600 \cdot N$ kb/s and minimum value $A_m = 100 \cdot N$ kb/s (where N is the number of total concurrent flows). Figure 7 shows the test results. In the case of 1 QT and 1 TCP flow, after each over-buffering phase, the QT flow takes a bandwidth roughly equal to B_{avg}, so that the TCP flow (which is a greedy source) can take a larger fraction of bandwidth. On the other hand, in the case of 2 QT and 2 TCP flows, the QT flows always take more bandwidth than TCP ones.

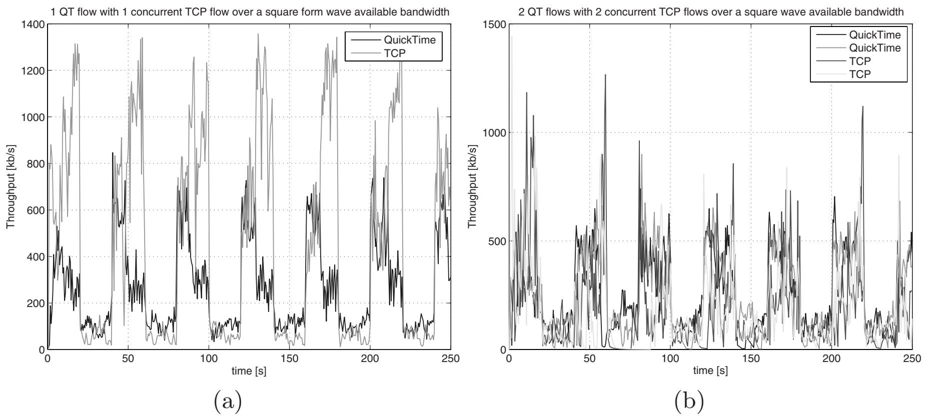


Fig. 7. QuickTime flows with concurrent TCP flows over a square wave available bandwidth

Table 2. DSS vs TCP test results showing average throughputs (TP), loss rates (LR) and Jain Fairness Indexes (JFI)

Experiment	Flow	TP (kb/s)	LR (kb/s)	LR%	JFI
1 QT + 3 TCP, const bw	QT	250.44	27.09	11	0.96
	TCP1	198.18	24.45	12	
	TCP2	150.50	23.94	16	
	TCP3	179.20	23.61	13	
2 QT + 2 TCP, const bw	QT1	278.82	22.52	8	0.81
	QT2	215.62	31.61	15	
	TCP1	87.70	18.49	21	
	TCP2	121.14	21.92	18	
3 QT + 1 TCP, const bw	QT1	249.55	27.32	11	0.87
	QT2	216.97	36.67	17	
	QT3	144.67	33.40	23	
	TCP	93.76	18.46	20	
1 QT + 1 TCP, variable bw	QT	262.96	15.65	6	0.93
	TCP	453.86	24.17	5	
2 QT + 2 TCP, variable bw	QT1	253.24	20.76	8	0.97
	QT2	252.64	20.28	8	
	TCP1	195.46	24.23	12	
	TCP2	172.51	23.55	14	

6 Conclusions

We have carried out an investigation of the Darwin Streaming Server (DSS) in order to evaluate the behaviour of Reliable UDP in the case of time varying network bandwidth and in the presence of multiple concurrent QuickTime and TCP flows.

Main results are: i) when a Reliable UDP-capable client is used, there is an effective adaptation to the network bandwidth; ii) when more concurrent QuickTime flows share the same link, the available bandwidth is not shared fairly, depending on the over-buffering phase adoption by each flow; iii) QuickTime flows are unfriendly with respect to concurrent TCP flows, since they experience a goodput much higher than TCP ones.

References

1. Jacobson, V., Karels, M.J.: Congestion avoidance and control. ACM SIGCOMM Computer Communication Review (January 1988)
2. Floyd, S., Fall, K.: Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking (May 3, 1999)
3. Ding, L., Goubran, R.: Assessment of effects of packet loss on speech quality in VoIP. In: Proc. HAVE 2003, pp. 49–54. IEEE, Los Alamitos (2003)
4. Hayashi, T., Yamasaki, S., Morita, N., Aida, H., Takeichi, M., Doi, N.: Effects of IP packet loss and picture frame reduction on MPEG1 subjective quality. In: IEEE 3rd Workshop on Multimedia Signal Processing 1999, pp. 515–520 (1999)

5. Handley, M., Floyd, S., Padhye, J., Widmer, J.: TCP Friendly Rate Control (TFRC): Protocol Specification. Proposed standard (January 2003)
6. Kohler, E., Handley, M., Floyd, S.: Designing DCCP: Congestion Control Without Reliability. Proposed standard (May 2003)
7. QuickTime Streaming Server Modules Programming Guide
8. Guo, L., Tan, E., Chen, S., Xiao, Z., Spatscheck, O., Zhang, X.: Delving into internet streaming media delivery: a quality and resource utilization perspective. In Proc. of ACM SIGCOMM IMC 2006, pp. 217–230 (2006)
9. Chung, J., Claypool, M., Zhu, Y.: Measurement of the Congestion Responsiveness of RealPlayer Streaming Video Over UDP. In Proc. of the Packet Video Workshop (PV) (2003)
10. Nichols, J., Claypool, M., Kinicki, R., Li, M.: Measurements of the congestion responsiveness of windows streaming media. In Proc. ACM NOSSDAV 2004, pp. 94–99 (2004)
11. Hessler, S., Welzl, M.: An Empirical Study of the Congestion Response of RealPlayer, Windows MediaPlayer and Quicktime. In: Proc. of IEEE ISCC 2005, pp. 591–596 (2005)
12. Andreasson, O.: Iptables Tutorial 1.2.0. World Wide Web (2005), http://iptables-tutorial.frozentux.net/iptables-tutorial.html
13. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Computer Communication Review 27(1), 31–41 (1997)
14. Hartwell, J.A., Fapojuwo, A.O.: Modeling and characterization of frame loss process in IEEE 802.11 wireless local area networks. In Proc. IEEE VTC 2004, vol. 6 (2004)
15. Chiu, D.M., Jain, R.: Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems 17(1), 1–14 (1989)

An Encryption-Enabled Network Protocol Accelerator

Steffen Peter, Mario Zessack, Frank Vater, Goran Panic, Horst Frankenfeldt, and Michael Methfessel

IHP GmbH, Im Technologiepark 25, 15236 Frankfurt/Oder, Germany
{peter,zessack,vater,panic,frankenfeldt,
methfessel}@ihp-microelectronics.com

Abstract. Even in light-weight wireless computing applications, processing of network-protocols becomes more and more computation- and energy-hungry, with increasing data rated and the need for security operations. To cope with such requirements and as alternative to heavy-weight computation systems we propose an embedded system that is build for fast network-processing while supporting acceleration of state-of-the-art symmetric (AES) and asymmetric (ECC) cryptographic operations. We demonstrate how to build a dedicated TCP accelerating system based on a profiling analysis. We also discuss optimized implementations of the AES and ECC cryptographic protocols while considering the trade-off between software and hardware. Compared to an initial software-only implementation our final system accelerates the protocol handling by a factor of three, while the cryptographic operations are improved by two orders of magnitude. Our system which was manufactured in 0.25μm CMOS technology needs about 55 mW for a data rate of 40 MBit/sec.

1 Motivation

Light-weight networked devices in the emerging world of ubiquitous computing must cope with ever-icreasing amounts of data. For instance, surveillance applications ranging from small sensor readings up to real time video require reliable, fast, and efficient network processing. Additionally, wireless and embedded devices are becoming integral parts of safety-critical and long-living systems, e.g., in applications to monitor buildings, cars etc. This implies use of strong security means to ensure data integrity and authenticity. Suitable encryption methods are very demanding computationally. Together, there is a need for efficient network processing in combination with state-of-the-art encryption methods. In this paper, we present results of hardware/software co-design to develop and implement a network processor with encryption capabilities.

Such a processor is useful in the context of computers with limited resources, such as laptops or PDAs. Another application is to enable "dumb" devices with secure network access. Specific applications are in the area wireless sensor networks (WSN). Sensor nodes usually cannot easily bridge from their internal network protocol to wide area networks. Such gateway tasks are typically performed by heavy-weight personal computers (PC). While this solves the technical problems, economically and practically this approach is often not viable. For example, in border-land protection the environment-observing nodes transfer their results to a cluster head which forwards the results to a control center. Such cluster-head nodes are required every 50 to 100 meters, making PCs unfeasible.

The cluster head also performs security control. An abundance of security protocols for WSNs have been proposed, but the most promising solutions require a trust center. The trust center observes the network behavior (intrusion detection) and does cryptographic operations and authentication. The battery and processing power of embedded devices is often not sufficient to run the required strong security algorithms. This underlines the need to equip light-weight devices with hardware accelerators, which increase the performance and reduce the energy consumption for cryptographic operations in combination with network processing.

This paper proposes an integrated solution for this purpose. Although we focus on the TCP/IP protocol, the results are applicable to other transport protocols as well. The solution includes cryptographic hardware accelerators that sufficiently improve the applicability of strong secure cryptographic algorithms. The discussed design is a concrete implementation, but the paper provides a general blueprint for light-weight, hardware-accelerated implementations which combine network protocol handling and cryptographic operations.

The rest of this paper is structured as follows. In Section 2 we profile a TCP implementation in order to determine the performance-critical operations. Then we evaluate potential solutions for the bottlenecks while referring to related work. On this basis we derive a general network accelerator design in Section 4. Potential solutions for fast cryptographic implementations are investigated in Section 5. In Sections 6 and 7 we discuss the implementation and the results before we conclude the paper.

2 Profiling TCP on an Embedded Processor

A first step toward an efficient implementation is to understand what the critical and time-consuming tasks are. For the TCP protocol, a comprehensive performance analysis has already been reported in [5]. The focus there is on computer systems with the Windows or Linux operating system. The results show that about 50% of the processing effort is kernel or driver-related. In single-thread-operating systems, as they are applied in embedded environments, we expect significantly different figures. For a reliable determination of these values we implemented a TCP/IP stack on an embedded system. We added hooks to measure the time spent in the various subroutines.

The implementation includes a fast path that processes incoming and outgoing data much faster when no special treatment is needed. If the data packets arrive in correct order without errors, there is no need to process the full conditional TCP logic. In our test cases, 5 Megabytes of data were transferred. We

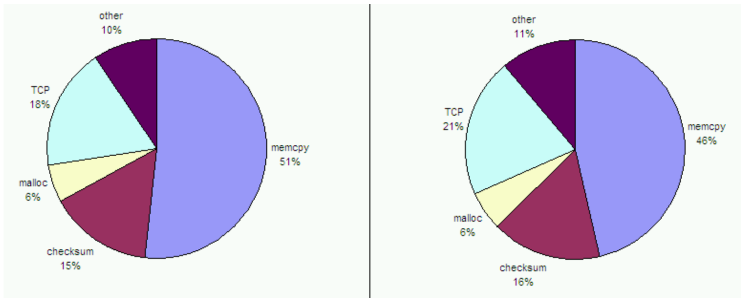


Fig. 1. Profiling results for transmitting (left) and receiving (right)

assumed a good connection that only passes the slow path to establish and close the connection. The 1815 data packets are entirely processed in the fast path. Assuming reasonably good transmission conditions in realistic cases, the obtained values show where to invest effort in order to improve the efficiency.

The total processing time shows that receiving needs slightly more effort than transmission (4.7 sec. for send, 4.9 sec. to receive). This is not particularly surprising and has already been reported in [1]. More relevant is the distribution of effort among different processing steps. An overview of the values for sending and receiving can be seen in Figure 1.

During transmission, most time is spent for copying data from the application memory to the TCP send buffer and the packet structure. Computation of the checksum and the final send operation also require significant processing time. During reception, the major operations are copying of the data from the network memory to the internal TCP structure, computation of the checksum and finally transfer of the data from the TCP memory to the application. On the other hand, the actual logic of the TCP protocol (the state machine, congestion control, and the conditional logic) does not need even 20% of the processing time.

3 Related Work

The profiling results published in [5] already indicate that the actual protocol processing and the checksum computation are not the major performance bottlenecks of a TCP implementation. Basically our results confirm this. Nevertheless, the two most discussed approaches in related work propose acceleration of these functions. The first approach of such offload engines [6] is to design the TCP state machine in hardware. However, with a software-implemented fast path for general data sending and receiving, the complicated state machine is bypassed 97% of the time. Thus the potential advantages of a hardware protocol state machine are minor. In contrast the potential disadvantages of a hard-coded protocol implementation - in particular the lack of flexibility - would clearly outweigh the benefits.

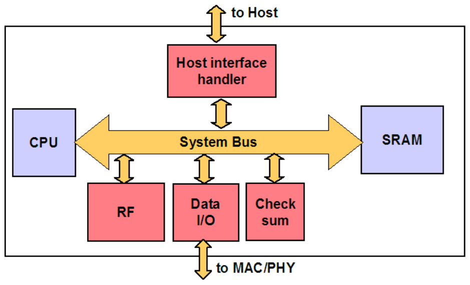


Fig. 2. General design of our network accelerator: Checksum can be computed while data is written into memory

The second well-known approach is the hardware implementation of the checksum operation. A hardware block to compute the TCP checksum is tiny (one 16 bit adder and a register) and silicon costs are negligible. However, our profiling results show that the potential performance gain, while not negligible, is not of primary importance. The checksum operation for both receiving and sending requires merely 16% of the total processing time.

Alternatives are onload implementations that implement the packet processing onto a dedicated set of computer resources. Usually this means dedicating a processor core. According to [10] this allows optimizations that are impossible when time sharing the same computer resources between the operating system and applications. Although the idea of a dedicated CPU core for network processing is interesting, it does not suit in our light-weight scenario.

Several offload and onload engines as well as hybrid approaches [13] have been proposed. However, these mostly focus on TCP server applications which have - as our profiling results already showed - different requirements and properties. Anyway they do not address the mainm bottleneck of our profiling. The most time consuming operation is copying. Since our tested implementation requires two copy operations per direction, the problem becomes more severe. The additional copy operation from and to the network adapter cannot be omitted. Single-copy or even zero-copy TCP stack implementations have been proposed to tackle the problem [14], but this typically compromises the socket API and implies undesirable non-standard handling in the user code.

One result of our profiling is that memory allocation does not require a notable amount of processing time, consuming about 6%. Each packet needs a memory slot and is stored in retransmit or receive buffers. On personal computers, memory management is a task for the operating system. Om embedded devices this task must be considered separately.

4 System Design

Based on the results of the profiling, we design a TCP offload device with the primary targets of reduced energy consumption and support of a data rate of

54 Mbit/sec, without yet considering encryption. For the power consumption, a value of 50 mW was targeted in the in-house 0.25μm CMOS Technology. To archive these goals, the main strategy is to reduce the CPU load by doing specific steps in hardware: a) copy operations, and b) evaluation of the checksum. Other parts of the TCP protocol are done in software on the embedded CPU. With this general strategy, we expect to reduce the utilization time of the CPU by more than 80% (in comparision to a reference solution which simply ports the protocol to the embedded processor) because only the protocol handling (<20%), memory allocation (5%) and additional control information are processed by the CPU. Our basic architecture is shown in Figure 2. The offload device consists of these components:

CPU: The CPU is a 32 bit MIPS processor.
Bus: The components are connected by a 32 bit AMBA bus [8].
SRAM: Packets are stored in a 32 kByte SRAM memory.
RF: A register file controls the operation of the system.
Checksum: The checksum is computed for the data transferred over the bus
Network device interface handler: The hardware unit copying data between the internal SRAM and the network device.
Host interface handler: A unit connecting to a host device

The design does not contain a separate memory allocation unit. Although the memory operations consume a noticeable slice of the total CPU time, we decided not to implement a dedicated memory manager hardware block. Tests showed that such a unit would accelerate the memory allocation by 77%, even considering the additional system communication overhead. However the the additional hardware costs (300 flip-flops for the 32kB SRAM) and the potential lack of flexibility deterred us from exercising this option.

Now consider the case that the host wants to send data. First, it would register the transmission, so that the embedded CPU creates a socket and allocates memory. The host would directly copy the data into the assigned memory region, whereby the checksum is computed. When a packet is full, the CPU finishes the header processing. Then the network device is given the address of the packet and finally transmits it on its own.

Incoming packets are also copied to an assigned memory area in the internal SRAM. During this copy operation the checksum is computed. If the incoming packet is received and not corrupted, the CPU is informed and starts to process the header. At all other times the CPU can sleep and thus reduce the energy consumption.

This design employs the CPU solely for complicated operations which are not of primary significance for the total effort, i.e.,

– Build-up and tear-down of a connection.
– Management of the state machine and sockets.
– Congestion control: adaption of the transmit rate to network load.
– Error handling: unexpected packets, retransmission etc.
– Software handling allows protocol variations and debugging.

One sees that the CPU is not involved in those steps which touch the data payload (copying and checksum). It operates only on packet headers and on internal data needed for book keeping. Since the amount of this data is small compared to typical packet payloads, the implementation is expected to be efficient.

As described, the system is driven by an external host, allowing the CPU to sleep a large percentage of the time. Alternatively, the system could be used as a stand-alone solution, as would be the case for a light-weight gateway in a WSN. Here the CPU would process the application code in the remaining time. In either case, the CPU could also be used for cryptographic operations, e.g. to encrypt the payload or to handle cryptographic protocols. In the next section, we consider how to add encryption in hardware to the design.

5 Cryptographic Functions

In this section we discuss implementations for two prototypical cryptographic protocols: AES and ECC. Both are standardized and considered as strongly secure. Since our goals are good performance and reduced energy consumption, we will discuss hardware accelerators and their potential trade-offs.

5.1 AES

The Advanced Encryption Standard (AES) is the replacement for the insecure DES-Algorithm. It was standardized in 2001 [16] by the NIST. It is a symmetric block cipher algorithm, which uses the same key for encryption and decryption. The data block length is 128 bit. In contrast to the data block length, various key lengths are possible. Three different key lengths are standardized. The shortest, and also the most often implemented version, has a length of 128 bit. Longer (but rarely used) key lengths are 192 and 256 bit. Especially for low area implementation those versions are not well suited. They require additional area for key registers and the runtime increases by up to 30%.

The AES algorithm itself consists of four parts in 10 rounds: key addition, bytewise substitution, shift in rows and mixing of the columns. The nature of every step allows an efficient implementation in hardware as well as in software.

The implementation applied in our system is similar to [17]. The standard implementation has a memory-like interface with a 4 bit address and a 32 bit data bus. First, the key is loaded into the key register. Next, the data to be encrypted/decrypted is loaded. The algorithm starts automatically after the last chunk of the four 32 bit data blocks is written. We have adapted the key management so that it is possible to select between two different stored keys. We used a full 128 bit wide key interface instead of the memory-like interface. Changing the key (e.g., to use different keys for encryption and decryption or for bidirectional communication) costs only a control word instead of a control word plus four times a data word (32 bit).

To optimize the implementation, we analyzed the requirements and the performance of a possible straight-forward AES implementation. We share the S-Box

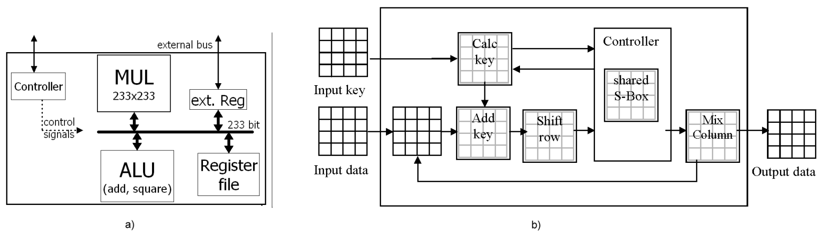


Fig. 3. Schematic of a) the ECC design and b) the AES accelerator

with the key generator and the algorithm itself Fig. 3 b). Furthermore we replaced the standard Mixcolumn function. Usually an independent component is used for each direction. We reuse the Mixcolumn function for encryption in the decryption path as mentioned in [18] what could save 44% of silicon area.

As alternative software implementation on our embedded CPU we simulated encryption and decryption with an optimized AES software implementation[2]. Table 1 shows the measured results for both software and hardware. The latter - independent of direction - requires 78 clock cycles including I/O-Operations. Assuming that the MIPS core requires 44mW at 33MHz a 1MByte block consumes 95.3mJ in power to encrypt the data. The AES core requires for the same data block 1.04mJ. Furthermore the hardware accelerator is about 67 times faster than the encryption in software and 89 times faster than decryption in software.

Table 1. Comparison of the hardware AES design to a MIPS software solution

		Clock cycles	Time [μs]	Energy per 1MByte [mJ]
En-	Hardware	78	2.5	1.04
cryption	Software	5228	172.5	95.30
De-	Hardware	78	2.5	1.04
cryption	Software	6857	226.3	125.00

5.2 ECC

Symmetric cryptographic approaches, like AES, are considered to be secure and computation costs are relatively low. However they do not always provide satisfying answers to questions regarding authentication, key distribution, and ensuring of data integrity. Here asymmetric approaches, also known as Public Key Cryptography (PKC), are a suitable solution. We focus on Elliptic Curve Cryptography (ECC) since it provides a good level of security even with relatively short key sizes, leading to relatively low calculation costs compared to other PKC-approaches. But, we are convinced that processing time and power consumption are still too high if all operations are executed in software on a light-weight device. This is why we propose a hardware design that can accelerate the ECC operations on our network processor.

Figure 3 a) depicts the block diagram of the exemplary 233 bit ECC hardware accelerator. The arithmetic units and register are connected by a 233 bit internal bus. The control unit manages the bus access and the operations. This is the place where the ECC algorithms are executed. In our design the elliptic curve point multiplication (ECPM) is performed by the Lopez-Dahab algorithm [9]. The control unit also manages the access to the eight 233 bit registers. One of these registers can additionally be written from the external bus.

The ALU combines the functionalities of addition, squaring and allows bit manipulations. The multiplier is an Iterative Karatsuba Multiplier, as proposed in [4]. It requires 9 cycles for each 233 bit operation. It is not only the largest unit but also the most utilized one. The duty time is more than 90%.

As for the AES we want to compare the hardware design to a software implementation on the embedded CPU executing the ECC operations. The ECC software implementation is based on the MIRACL library [12] and was run on the system without utilizing the crypto-accelerators. A point multiplication on the curve B-233 takes 13 million clock cycles which corresponds to 400 ms. The code size for this implementation is 48 kilobytes. An alternative implementation requires only 14 kilobytes but is much slower with 900 ms required for a point multiplication. The code size must be considered when memory is an issue, as it is for many small mobile devices.

Table 2 shows a comparison of the 233 bit MIRACL software implementation with the ECC hardware design as both are implemented and running on the communication SoC. All the data were measured in the simulation environment for one 233 bit point multiplication at a speed of 33MHz. We used the simulation environment in order to isolate the power consumption for the operation. The results were verified on the actual hardware after manufacturing the chip.

Table 2. Comparison of the 233 bit ECC hardware design to a software solution on the SoC

	Time[ms]	Power[mW]	Energy[mWs]
software	410.2	40.2	16.490
hardware	0.4	75.6	0.030

The results show that the hardware solution is 1000 times faster and consumes 550 times less energy in comparison to the software implementation.

It should be mentioned that efficient assembler language supported software implementations can give better performance than the chosen MIRACL library. For example the StrongARM (206 MHz) implementation presented in [11] needs 9 ms and less than 4 mWs. It still is significantly slower than the hardware design and needs two orders of magnitude more energy per ECC-operation.

5.3 Integration of the Crypto-Accelerators in the SoC

Both cryptographic accelerators were integrated into the network accelerator. The AES is suitable to protect the transmitted payload. The more expensive public key approach ECC is used for key establishment and for authentication.

The two different areas of application impact our integration decision. The AES module is located before the internal SRAM. When enabled, it transparently encrypts (or decrypts) the data flowing into or out of the packet memory. In this way, the initial copy operation can fill and encrypt the packet payload, and compute the checksum over the encrypted data. All these operations are executed transparently within one operation.

Since ECC is not used to encrypt payload data, we decided to connect it directly to the system bus. The CPU (but also an external host) can address the registers of the ECC accelerator and transmit data and keys. While the ECC unit is working, the CPU can perform other tasks or sleep.

6 Implementation of the TCP/Encryption Chip

First, the general work flow is described. All functional blocks (excepting the MIPS core, the AMBA bus, and SRAM) were implemented in VHDL. Behavioural simulations of the individual VHDL blocks verified the the implementations. Comprehensive hardware/software co-simulations of the complete system with the embedded TCP stack showed the correctness of the entire design and provided first estimates of the performance and energy consumption. In addition, we ported the design to an FPGA in order to test the system in the the real world at real speed.

Our primary target was an implementation in silicon. The ASIC was synthesized with the library of our in-house $0.25\mu m$ CMOS Technology[7]. The resulting netlists with detailed timing information were the basis for the calculation of the power consumption using Synopsys PrimePower[15]. This provides a precise gate level power estimate based on the the technology library and realistic test pattern, since the real transitions for the calculation are considered instead of merely statistical assumptions.

Since our goal was to test the concepts in different environments, we added various input/output interfaces (UART, GPIO, SPI). The host computer is connected via a CardBus interface. The connection to the network device is either per UART or EPP interface. The current prototype requires a rather large number of pins (256, of which 219 are signal pins). The large number is due to the CardBus interface and the 32 bit bus to the external memory.

A chip photo can be seen as Figure 4. The pads and the internal SRAM and cache (total 56 kByte) determine the total core size (54mm^2). A dedicated application specific design would not need all components and interfaces, and thus reduce the total area significantly. Also, a state-of-the-art 0.13 μm CMOS technology would additionally reduce the size for the chip.

For the results discussed in the next section, the open-source lwIP TCP/IP protocol stack [3] was ported to our system, extended by software memory

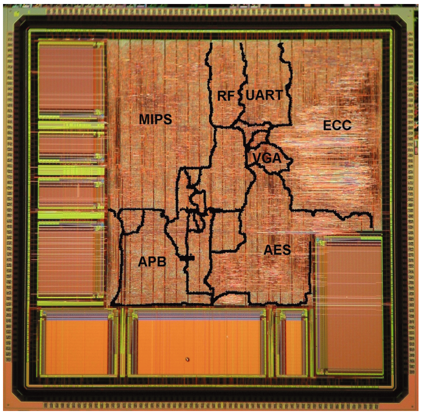


Fig. 4. The final system on chip. We also marked the largest internal blocks. APB is the bus and RF is the central register file. The rectangle blocks are memories.

management functions malloc and free. No operating system was used. The CPU executes one task at a time, and goes into a power-saving sleep mode when done. The sleep mode is left when an interrupt is given by the register file, which controls the overall system.

7 Results and Discussion

Our goal was to build a network accelerator that reduces power consumption and increases the network performance. Our approach implements dedicated hardware accelerator blocks while the executing complicated protocol operations in software on the embedded processor. Here we answer the main questions: what we actually have gained by the hardware support, and whether the original goals were met. For this purpose we estimated power consumption and attained data rate for three cases:

1. (SW) Pure software implementation on the embedded MIPS processor, at maximal data rate attainable.
2. (HW 1) With hardware accelerators, at the same data rate as (SW).
3. (HW 2) With hardware accelerators, at the maximal data rate attainable.

All values correspond to a clock frequency of 33.3 MHz, which is the clock supplied by the CardBus interface.

Table 3 splits the consumed power among the system blocks, based on the PrimePower simulations. Measurement of the total consumed power for the actual chip gives somewhat larger values. One reason is that the power consumed by the pads is included in the measurement. The table shows that the hardware design indeed increases the attainable data rate and reduces the energy consumption significantly. The maximal data rate in pure software on the MIPS

Table 3. Results of performance and power consumption of three scenarios

Case	Rate (Mb/sec)	CPU active	CPU (mW)	Bus (mW)	Regs (mW)	I/O (mW)	Total power
SW	20.7	100%	60	14	7	8	89 mW
HW 1	20.7	15%	9	14	7	12	42 mW
HW 2	40.0	31%	18	14	7	16	55 mW

CPU is 20.7 Mbit/sec. In this case the CPU works 100% of the time, by construction. The CPU (including cache) then requires about 70% of the total power (60 mW). The goal of 54 Mbit/sec can not be reached by the pure software version.

The second configuration (HW 1) runs at the same speed as is attainable by the software implementation. By comparing these cases, we can identify the power saving due to the hardware blocks. The CPU is now utilized for only 15% of the time. Thus, the hardware accelerators reduce load on the CPU and thereby save more than 50% of the total power. However, we don't quite save the 80% expected from the profiling. Indeed, the power consumed by the CPU does decrease by 85%, offset by a small increase due to the hardware I/O. It is the overhead due to other blocks such as the system bus, register file, and I/O which becomes a significant portion. The power consumption of I/O, register and bus are roughly at the same level as the CPU. Thus further optimizations are not as easy, since no single block needs more than 15% of the energy. The third row shows the data rate and power consumption for the maximal data rate attainable with the implemented system. With 40 Mbit per second it is lies below the target of 54 Mbit/sec. Our investigations identified that the current CardBus host interface does not allow a higher data rate (without using burst mode). The system design itself would allow more than 100 Mbit per second at 33 MHz clock frequency. At the maximum data rate for the manufactured system of 40 Mbit/sec, the CPU is busy for 31% of the time, leaving 70% for an embedded application.

For the encryption blocks, the performance was verified for the manufactured system. From the PrimePower analysis, the consumed power for the AES block is 17 mW when idle and 52 mW when active. For the ECC unit, the idle/active power is 25 mW and 60 mW. For simplicity, these contributions were omitted in the discussion of the power consumption for protocol handling above. The relatively high idle values for the encryption units shows the necessity of means such as clock gating to eliminate the power used by idle parts of the design.

Overall, results for the TCP protocol processor are close to the originally targeted values, but do not quite reach them. The gain by hardware accelerators for the protocol processing is not as spectacular compared to the accelerator blocks of the cryptographic operation, where we could improve the results by two orders of magnitude. For protocol processing the advantage factor of the network acceleration for performance and power consumption are five and two , respectively.

8 Conclusions

This paper describes the design and implementation of an offload protocol processor which can efficiently handle TCP protocol processing together with encryption by AES or ECC in hardware. By profiling of a software TCP implementation, copying of data payloads and (to a lesser extend) evaluation of the TCP checksum were identified as suitable candidates for hardware acceleration, i.e., the processing steps which pass over the data payload. Operations on the packet headers and for TCP bookkeeping consume only a small percentage of the total power, and are best done in software on an embedded CPU, permitting more flexibility and avoiding the development of dedicated hardware units. The system designed along these lines was further enhanced by adding hardware units to perform AES and ECC encryption. Since AES was intended to encrypt the data payloads, this unit was placed in the data flow before the internal SRAM. The ECC unit was treated as an independent entity, communicating with the rest of the system via registers. This allows the lengthy ECC operations to be done in parallel to other operations on the CPU. The final design was manufactured using the IHP in-house 0.25μ CMOS technology.

A combination of measurements on the manufactured chips and detailed simulations of the power consumption determined the performance of the system, splitting the consumed power by the different design units. As a basis for comparison, the TCP protocol was also done in software on the embedded CPU. In this case, a data rate of 20.7 Mbit/second is reached at a consumption of 89 mW, with the CPU is running continuously. If the hardware accelerators are used for this data rate, the utilization of the CPU drops to 15% and the power to 42 mW. A large part of the power now is consumed by the system bus, the register file, and I/O. The maximal attainable data rate is 40 Mbit/sec at a power consumption of 55 mW. Here, the CPU is still only active 31% of the time, since the rate is limited by the CardBus interface to the host.

A number measures could be implemented in an improved design. First, better solutions should be sought for units such as the system bus or register file, since these dominate the power consumption when the TCP hardware accelerators are used. Second, a more efficient way to move data between the host and the system such as DMA should be used. Third, clock gating should eliminate the power consumption by blocks which are idle, notably the encryption units, when these are not actively used.

Overall, the presented system is an efficient implementation of a combined protocol/encryption processor, which could be further improved by measures suggested by the evaluation of the manufactured system.

Acknowledgment

This work was partially funded by the German Ministry of Education and Research under grant 01AK060B.

References

1. Clark, D.D., Jacobson, V., Rornkey, J., Salwen, H.: An analysis of tcp processing overhead. IEEE Communications Magazine, 23–29 (1989)
2. Daemon, J.: AES implementation, optimized ANSI C v2.0., http://www.iaik.tugraz.at/research/krypto/AES/old/~rijmen/rijndael
3. Dunkels, A.: lwIP – a lightweight TCP/IP stack (October 2002), http://www.sics.se/~adam/lwip/
4. Dyka, Z., Langendoerfer, P.: Area efficient hardware implementation of elliptic curve cryptography by iteratively applying karatsuba's method. In: DATE, pp. 70–75 (2005)
5. Foong, A.P., Huff, T.R., Hum, H.H., Patwardhan, J.R., Regnier, G.J.: Tcp performance re-visited. In: ISPASS 2003: Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software (2003)
6. Freimuth, D., Hu, E., LaVoie, J., Mraz, R., Nahum, E., Pradhan, P., Tracey, J.: Server network scalability and tcp offload. In: ATEC 2005: Proceedings of the USENIX Annual Technical Conference 2005 (2005)
7. Innovations for High Performance microelectronics. IHP microelectronics: technology (2006), http://www.ihp-ffo.de/24.0.html
8. ARM Limited. AMBA specification, revision 2.0 (1999), ARM website http://www.arm.com
9. López, J., Dahab, R.: Fast Multiplication on Elliptic Curves over $GF(2_{\mathrm{m}})$ without Precomputation. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, Springer, Heidelberg (1999)
10. Regnier, G., Minturn, D., McAlpine, G., Saletore, V., Foong, A.: Eta: Experience with an intel xeon processor as a packet processing engine. In: 11th Symposium on High Performance Interconnects (2003)
11. Riedel, I.: Security in ad-hoc networks: Protocols and elliptic curve cryptography on an embedded platform. Master's thesis, Ruhr-Universitaet Bochum (2003)
12. Scott, M.: MIRACL—A Multiprecision Integer and Rational Arithmetic C/C++ Library, Version 5.0. In: Shamus Software Ltd, Dublin, Ireland (2005), http://indigo.ie/~mscott
13. Shalev, L., Makhervaks, V., Machulsky, Z., Biran, G., Satran, J., Ben-Yehuda, M., Shimony, I.: Loosely coupled tcp acceleration architecture. In: HOTI 2006: Proceedings of the 14th IEEE Symposium on High-Performance Interconnects (2006)
14. Steenkiste, P.: Design, implementation, and evaluation of a single-copy protocol stack. Software Practice and Experience 28(7), 749–772 (1998)
15. Synopsys Inc. PrimePower: Full-Chip Dynamic Power Analysis for Multimillion-Gate Designs (2005), http://www.synopsys.com/products/power/primepower_ds.pdf
16. FIPS U.S. Department of Commerce/NIST. Advanced Encryption Standard (AES), FIPS PUB 197 (2001), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
17. Vater, F., Langendörfer, P.: An area efficient realisation of aes for wireless devices. it - Information Technology 49(3), 188–193 (2007)
18. Wolkerstorfer, J.: An asic implementation of the aes-mixcolumn operation. In: Austrochip (2001)

Freemote Emulator: A Lightweight and Visual Java Emulator for WSN*

Timothée Maret[1], Raphaël Kummer[2], Peter Kropf[2], and Jean-Frédéric Wagen[1]

[1] TIC Institute, University of Applied Science of Fribourg,
Bd de Pérolles 80, CP 32, CH-1705 Fribourg, Switzerland
{timothee.maret,jean-frederic.wagen}@hefr.ch
[2] Computer Science Department, University of Neuchâtel,
Emile-Argand 11, CP 158, CH-2009 Neuchâtel, Switzerland
{raphael.kummer,peter.kropf}@unine.ch

Abstract. Research on Wireless Sensor Networks (*WSNs*) has developed highly optimized software environments fitting the limited hardware resource constraints of Motes. Unfortunately, these environments suffer from relatively complex programming models. Nowadays well known languages such as Java and optimized JVMs become available and simplify the application development for the Motes. Thus, we developed the Freemote Emulator which is a Java based emulator providing a lightweight emulation tool for emerging Java based Motes. It runs experiments in real time mixing real and emulated nodes. Its layered architecture and a set of predefined code templates allow developers to quickly produce runnable code for real and emulated nodes as well as predefined scenarios to help the newcomers to introduce into the system and WSNs. Our emulator provides as well a useful visualization tool based on a parametrizable slow down feature that helps to understand complex WSN behaviours and to debug tricky implementation problems. Finally, a single emulation can run on several computers, thus allowing programmers to conduct experiments with a pretty large number of emulated and real nodes.

Keywords: Wireless Sensor Networks, Lightweight Emulator, Java Based Motes, Freemote.

1 Introduction

Research on software tools and applications for Wireless Sensor Networks (WSN) is much guided by the hardware constraints of Motes such as a small memory footprint, limited energy and computational power. Operating systems like TinyOS running on such components are specialized to work with these constraints but suffer from complex and hard to learn programming models and languages. Some research has been carried out to produce virtual machines

* The work presented in this paper was partially funded by the HES-SO projects No 17223-ADS (development of the JMote) and No 16197-RASMAS (Freemote Emulator integration).

that run on top of TinyOS as for example Maté[13] or SwissQM[12]. These virtual machines provide simpler and more accurate programming interfaces. They define user extensible bytecode supporting different specialized programming languages. Unfortunately, they mostly remain "application specific virtual machines" as defined in [14].

Further optimized virtual machines for well known, high level and general purpose languages like Java have recently emerged which decrease the development time and complexity. The VM* framework[4] supports among other a subset of the Java language. Squawk[15] and Sentilla Point[25] are two Java virtual machines optimized either for heavily limited devices such as Java Cards or more powerful devices such as the Sun SPOT[24] platform.

In this paper we present the Freemote Emulator, a novel lightweight and distributed Java based emulator which aims at providing an emulation tool for the emerging Java based Motes. Rather than on performance evaluation accuracy, this emulator focuses on behaviour credibility by mixing emulated nodes possibly distributed on many networked computers and real nodes reachable through a specialized bridge. The Freemote Emulator divides the software architecture of a Mote in three independent layers connected through well defined interfaces: *Application*, *Routing* and *Data Link and Physical*. A unique XML file modifiable thanks to a user friendly GUI permits to dynamically select and configure the proper layer implementation.

Moreover, the development of specialized code for the real nodes is not necessary while the code of *Application* and *Routing* layers can be directly run on them with no adaptation. Currently the code runs on the JMote, a Java programmable Mote prototype developed at University of Applied Science of Fribourg. These Motes are based on a IEEE 802.15.4 compliant radio chip and support the same message format (Active Message) as TinyOS. Consequently, they are compatible with well known Motes and can perform in a heterogeneous environment thus increasing the reality of the conducted experiments. Additionally, the emulator supports the nodes playing different roles in the network and thus providing high flexibility in the scenarios emulated.

In addition, an optional network visualization tool displays the emulated nodes, the bridge node as well as the content of the messages sent and the physical topology. The emulator provides an interesting and tunable slow down feature that allows the developer to reduce the emulation speed to easily analyse the behaviour of algorithms.

Finally, as our emulator has been developed in Java, it can be quickly and easily started from a website with a set of selectable and predefined scenarios using the Java Web Start technology. These features place the Freemote Emulator as an attractive educational tool for students and newcomers in the sensor network field. It is also an interesting scientific tool because it is highly configurable and related to the reality through the real node integration.

The remainder of this paper is organized as follows. Section 2 discusses related work in the field of sensor network evaluation tools. The Section 3 describes the design and implementation of the Freemote Emulator. Section 4 presents an evaluation of the Freemote Emulator based on the implementation of a simple scenario. Finally, Section 5 discusses the design choices, concludes and lists the possible evolution of the Freemote Emulator.

2 Related Work

Researchers in the field of Wireless Sensors Network can choose among a large set of environments and tools to test and validate their developments ranging from real testbeds to less accurate application oriented simulators. A survey of more than forty tools for wireless networks can be found in [18]. Each tool has advantages and drawbacks and should be chosen depending on the experiment to be conducted in order to provide adequate results. Most of them target Berkeley Motes platforms and TinyOS applications, if they do not target abstract Motes and use a code translation phase to provide a platform specific binary code.

Mobile Emulab[1] and Motelab[11] are testbeds that provide time shared remote access to dynamically programmable mobile or fixed network of Berkeley Motes. The experiments done with these tools are close to reality, but deal only with a relatively small number of nodes.

ATEMU[5], Avrora[6] and MSPsim[7] are "fine-grain simulators" that operate at instruction level by simulating the Berkeley Motes processor instruction sets such as the MSP430 or the AVR family. As they care about the hardware particularities, these simulators provide highly accurate evaluation results of experiments including timing or power consumption aspects. However, they require much computational power and provide poor visualization tools. TOSSIM[9] is a step by step discrete event simulator for TinyOS applications which can be coupled to TinyViz to provide an extensible visualization tool. It runs the applicative nesC code unmodified and simulates the TinyOS behavior of the components tied to hardware.

The EMStar[10], SENS[8] and COOJA[2] environments are more similar to our work than the previous ones, as they provide a less accurate low level and radio simulation model. Moreover, they focus on network behavior analysis more than on time based performance evaluations.

Similar to our work, SENS is a layered and modular environment that runs applications (composed of interchangeable modules) written in C++, a high level and general purpose language. It differs from our environment as the code is not directly executable on real nodes but can be ported to Berkeley Motes. Moreover it does not support any bridge to real nodes. EMStar is an environment for testing applications for wireless networks which runs on Linux Microservers. EMStar provides the EmTOS facility that enables execution of TinyOS applications written in nesC on Microservers. This environment is highly versatile as it can mix Microservers and emulated Motes in the same experiment and can provide either simulated or real radio channels between emulated nodes based on an array of Berkeley Motes. Finally COOJA is a Java based simulator for the Contiki OS, able to mix simulated nodes at different level of detail in the same experiment. It is able to simulate Java code for prototyping. However the written Java code is not executable on real node but must be adapted or ported. This simulator also does not provide any bridge to real nodes.

3 The Freemote Emulator

3.1 Heterogeneous Experiments

The Freemote Emulator performs experiments that involve both real nodes and emulated nodes connected through a bridge node in real time. Fig. 1 represents a typical deployment of an experiment. The real nodes could be any Java JME CLDC1.0 or more powerful Motes based on the IEEE 802.15.4 LR-WPAN radio standard. A new suitable Mote such as the Sun SPOT (CLDC1.1) must simply provide an implementation of the Data Link and Physical layer interface to be adapted to the Freemote Emulator. This work is relatively simple if the basic sending and receiving services are already provided.

The present version of the emulator supports the JMote platform [20,21] developed at University of Applied Science of Fribourg under the Ad hoc Design Studio Project [19]. The JMote is based on the CC2420 radio chip like other known Motes (e.g., MICAz, TelosB). than for flexibility as they have 2MB of memory and a microprocessor supporting direct Java bytecode execution without needs of software virtual machine.

The bridge node is composed of an emulated and a real part. The real part is based on a MICAz or TelosB Mote that executes a code simply forward the

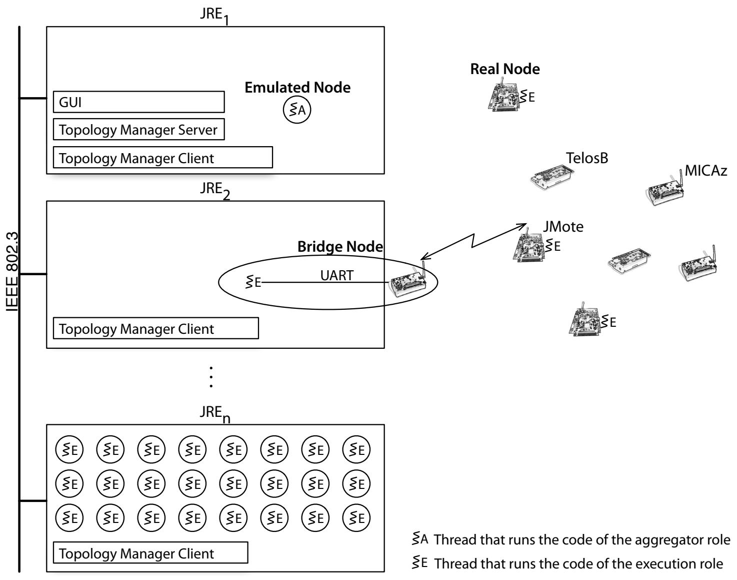


Fig. 1. Typical Deployment of an Experiment

messages from the radio interface to the UART and vice versa. The emulated part uses the TinyOS Java facility to send and receive messages from the UART. Fig. 1 shows that many processes (JRE *Java Runtime Environment*) to emulate nodes can be involved in the same experiment . These processes are inter-connected by an IEEE 802.3 network and can either be distributed over multiple computers or run concurrently on the same machine. Each process emulates one or more nodes, each one in a separate thread. The code for all the nodes emulated in the same process is the same but can change between processes. This feature permits to involve nodes with different roles (eg. aggregator, execution node) in the same experiment. The physical topology as well as the motion of the emulated nodes is computed by a single *Topology Manager Server* instance which provides this information to emulated nodes through *Topology Manager Client* instances. Each process contains a unique instance of the *Topology Manager Client.*The first process started in an experiment will also automatically run the *Topology Manager Server* and the GUI which displays the emulated nodes and allows to manage the evaluation.

3.2 Layered Architecture

Fig. 2 shows the layered architecture of the emulator for the three kinds of nodes that an experiment can involve. The proper implementation for the three layers is dynamically loaded with the settings defined in the XML configuration file. The multiplexer/demultiplexers define the interfaces between the layers as well as the methods to send and receive messages. Every messages sent at the *Data Link and Physical* layer or *Routing* layer is associated with an 8 bit label. This label is typically used upon reception to differentiate one type of message from another one. At the lowest layer, this information is directly mapped to the *type* field of the TinyOS Active Message, while a new field is added for the routing layer. An application that wants to receive messages must register as listener for the corresponding label on the multiplexer/demultiplexer. The lowest layer must implement an optional promiscuous mode which permits to receive every message sent by the physical neighbors, even if the destination of the message is not the current node.

The actual version provides some examples of applications such as the Ping like test, which is further detailed in Section 4, and a routing protocol based on AODV[22]. This implementation is optimized for the IEEE 802.15.4 standard as it uses the MAC level acknowledgements to detect the broken routes. The implementations of the lowest layer are tightly coupled to the kind of platform and are further detailed in the next section.

3.3 Data Link and Physical Layer Simulation

The *Data Link and Physical* layer is the only layer that has a different implementation for each kind of nodes. For the real nodes, most of the features are provided by the CC2420 radio chip, so the implementation simply contains a driver for this chip and a medium access control protocol similar to CSMA/CA.

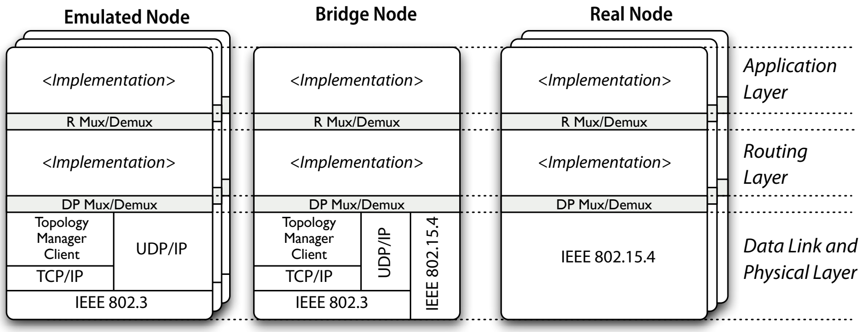


Fig. 2. Layered View of the Software Architecture

The emulated nodes and bridge node simulate the IEEE 802.15.4 behavior in a lightweight manner. In order to simulate the broadcast characteristic of the radio medium, a node must know its physical neighborhood before sending a message. This functionality is provided by the *Topology Manager* organised in a client/server manner. The clients act as caches and contain topological information only for the nodes emulated in the associated process. The server maintains a cartesian square map that contains all the emulated nodes and periodically updates their positions following the mobility scheme selected in the configuration file.

Different mobility schema are provided from mostly random to totally predefined schema. The latter is useful to describe repeatable scenarios. Once this update has been completed, the server computes the new physical topology. The algorithm in the *Topology Manager* first sorts the nodes following the X axis and then creates all the physical bindings in one run through the list of nodes. It operates in $\mathbf{O}(N(\log N + C))$ time complexity, where C is the average connectivity of the network and N is the number of emulated nodes. The bindings are unidirectional as each emulated node can have its own circular radio range. During this computation, the server proactively sends the topological modifications (add or remove a neighbor) to the clients using TCP sockets. This process guarantees that the clients contain always up to date topological informations.

Every time an emulated node needs to send a message, it first picks a random value. If this value is lower than the message error rate specified in the configuration file, and if the destination address is not the broadcast address, the message is discarded. This allows to take into account the effect of poor radio transmissions. In the other cases, the node fetches its list of physical neighbors from the client and then sends a copy of the message to each of them using UDP sockets. The sending node also checks that the destination of the message is in its neighborhood and if an acknowledgement is requested, it checks that the destination node is able to send back an acknowledgement (if it contains the sending node in its neighborhood). This simulates the IEEE 802.15.4 standard efficiently by avoiding the sending of acknowledgement messages and simply simulates the radio channel with a message error rate. However, the actual implementation of this layer does not provide a simulation of the channel collision detection which implies that two physical neighbors can send messages simultaneously without generating any error. Furthermore, it does not take into account the radio propagation delays, and considers neither the throughput nor the path loss and fading. The implementation of the *Bridge node* layer is similar except that it also sends the message to real nodes.

3.4 Simple Development Environment Based on Templates

The implementation of each layer can be easily extended as the Freemote Emulator provides templates of codes. The configuration GUI dynamically lists the selectable implementations by looking up specific package for each layer. If a developer wants to implement his own routing protocol or physical layer he can simply create a new class that implements the abstract classes provided for each layer. These classes already provide an access to the lowest layer implementations selected in the configuration file. We present below the process of building a new routing protocol implementation. This process is similar for the upper layer.

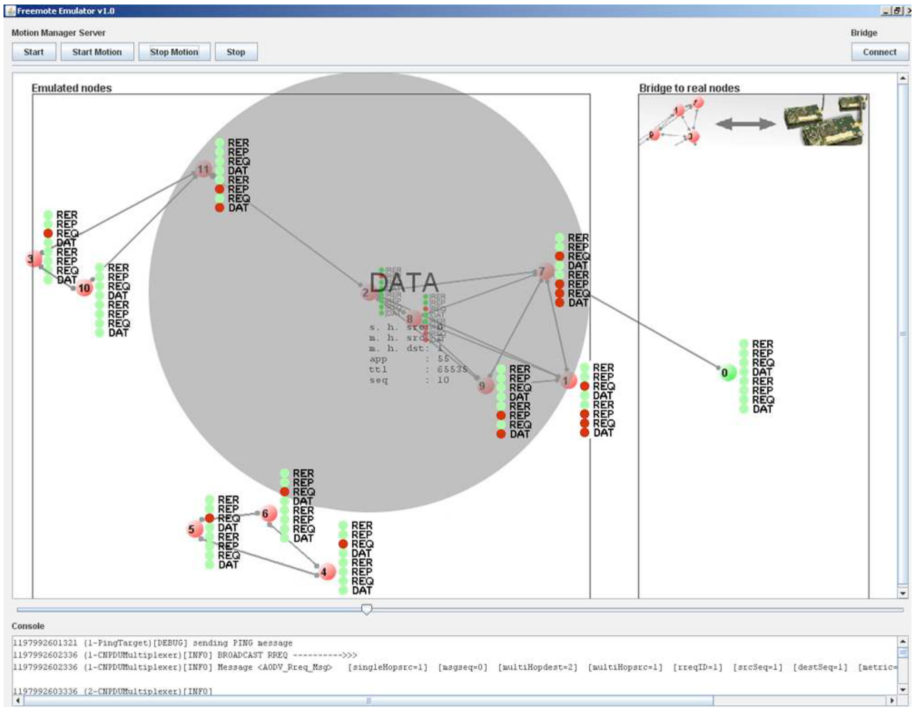
First, the `run` method (from the `Runnable` interface) must be implemented with the desired business logic. This method is automatically called by the environment at the end of the launching phase. The implementation should start by calling the following method in order to subscribe to the labeled messages exchanged at this layer: `dlpLayerSubscribeToLabel (byte label)`

Then the implementation should call the following method in order to build the protocol behavior: `dlpLayerSendMsg (byte label, I802_15_4_MPDU msg)`

Finally the following method must be implemented with the business logic code that processes the incoming messages. This method is automatically called by the environment upon reception: `dlpLayerProcessIncomingMsg (byte label, I802_15_4_MPDU msg)`

3.5 Powerful Visualization Tool

As shown in Fig. 3, the Freemote Emulator provides a powerful visualization tool that displays in real time a map containing the *Emulated nodes* and the *Bridge node*. The position of the nodes in the map points to the position simulated by the *Topology Manager* and is updated to visualize the motion of the nodes. The physical neighbor bindings are displayed by a grey line ending with a small circle that shows the binding direction. When a node sends a message, the developer can implicitly call a method that displays the content of the message and the radio coverage on the map. Moreover, this method will toggle one of the 8 LEDs associated with the node. This is visible on the JMotes and on the emulated Motes. The latter also display a small text near the LEDs that help to understand their meaning. This method slows down the execution speed by putting the current thread to sleep to give enough time to understand the network behavior. This is a key feature of the Freemote Emulator. The slow down factor is parametrizable in the configuration file. The standard output is redirected to the console in the GUI which logs the messages from every layer. The logging mechanism is organized with different levels identical to those defined by the framework Log4J[3] which permits to provide fine grained and context aware logs. Each layer can be parametrized independently from the others concerning the levels of displayed logs. This console helps to understand the behavior of the network as it maintains the history of the events and can display a large quantity of details.



```
1197992601321 (1-PingTarget)[DEBUG] sending PING message
1197992602336 (1-CNPDUMultiplexer)[INFO] BROADCAST RREQ ---------->>>
1197992602336 (1-CNPDUMultiplexer)[INFO] Message <AODV_Rreq_Msg>  [singleHopsrc=1]  [msgseq=0]  [multiHopdest=2]  [multiHopsrc=1]  [rreqID=1]  [srcSeq=1]  [destSeq=1]  [metric=

1197992603336 (2-CNPDUMultiplexer)[INFO]
```

Fig. 3. The GUI with the emulated nodes visualization tool

4 Evaluation

As an evaluation example of the Freemote Emulator we describe an experiment that runs a simple Ping like application on a highly heterogeneous network composed of emulated nodes, JMotes, MICAz and TelosB Motes. One emulated node has the role of an aggregator and runs the `PingAggregator` application that logs the evaluation data received from the execution nodes. The latter run the `PingEvaluation` application and some of them (listed in the configuration file) generate ping request. The Berkeley Motes are used only as routing nodes.

They execute a modified version of NST-AODV[23], a nesC implementation of AODV for TinyOS.

This scenario does not focus on the percentage of successfull ping request nor on timing aspects such as the round trip time. Indeed, these kind of results depend heavily on many parameters settings at each layer and on the motion of the nodes. However this scenario should demonstrate that the messages will make a successfully trip through every kind of node and that the reasons some messages are not conveyed are understandable with the visual information provided by the Freemote Emulator.

4.1 Simple Ping Scenario

In this scenario, only the emulated node with the address 1 periodically generates ping requests for the real node with the address 9 (see also Fig. 4). The requests must cover the network from emulated to real nodes. The emulated nodes move following a random walk mobility model for which the nodes periodically update their position by choosing a new random angle in $[0; 2\pi[$ and a new random speed in [0.5ms; 2.0ms]. The real nodes are positioned in a linear topology and will not move during the scenario. In order to run this experiment on a desk, the power of the Motes is set to the minimal value.

Fig. 4 shows a topology that could come up during the experiment. The average connectivity for the emulated nodes is set to 7, the message error rate is fixed to 3% and every emulated node has the same radio range. The logging mechanism is set in order to display all the logs of the routing layer. The slow down feature is set to 5 sec and we inserted a display instruction in the AODV implementation in order to visualize every message sent or received by the protocol. As this protocol uses four types of messages, we have associated each LED to a couple composed of the type of message and its direction (reception or sending). A LED toggles only if the corresponding message is received or successfully sent (reception of an acknowledgment, if requested). The real part of the bridge is a Crossbow Ethernet programming base MIB600 associated with a MICAz Mote. The main parameters of AODV are set as follows : the route life time is set to a value greater than the scenario execution in order for the routes to never expire; and the feature that lets intermediate nodes to repare broken routes is disabled.

4.2 Discussion and Results

Fig. 5 shows the environment in action during the execution of the Ping scenario. After ten minutes of observation of the network behavior based on the visualization facilities, the following results can be observed. First of all, between 70% and 80% of the requests reach the destination node 9 and successfully return to node 1 depending on the emulated nodes motion. As the real nodes are linearly arranged and the destination is at the end, the messages must pass through each kind of real nodes. This demonstrates that the JMote messages and those supported by the emulated nodes are compatibles with the Active Messages of TinyOS. Secondly, the visual information given by the 8 LEDs on every node

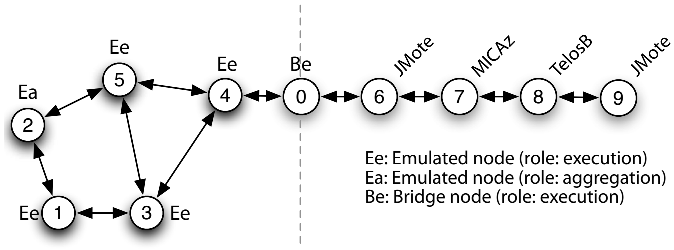

Ea
2
Ee
5
Ee
4
Be
0
JMote
6
MICAz
7
TelosB
8
JMote
9
Ee 1
3 Ee
Ee: Emulated node (role: execution)
Ea: Emulated node (role: aggregation)
Be: Bridge node (role: execution)

Fig. 4. Topology for the Ping scenario

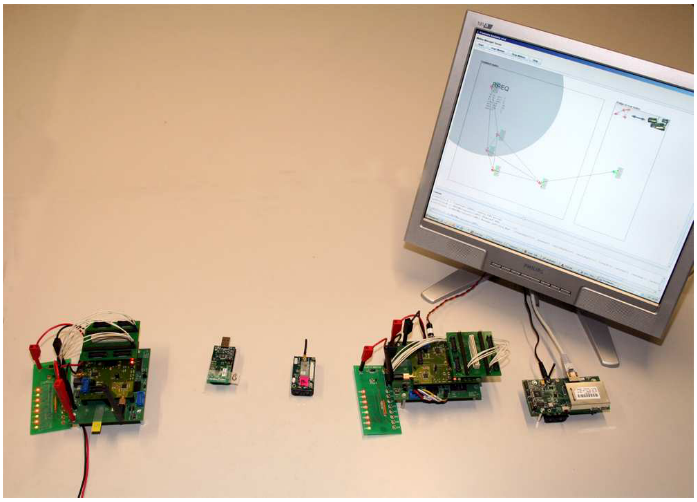


Fig. 5. The environment during the experiment. From left to right: JMote 9; TelosB 8; MICAz 7; JMote 6; Bridge 0 (real part); Emulated nodes 1 to 5 on the screen

is sufficient to easily understand the basic behavior of the AODV protocol such as the route construction by flooding, the construction of the direct and reverse path, and the generation of error messages to alert the originator node when a route is broken. However, to understand more complex behaviors, the set of LEDs is not expressive enough, but the emulated nodes provide more information. By looking at the content of the messages exchanged, the experimenter can easily understand the management of the sequence number or the method to choose the shortest route. The console contains all the information provided by displaying the messages as it keeps the whole history of events. However, the console has the drawback for the experimenter that he must look for the information in a huge quantity of data.

We have presented two visual and a textual feature provided by the Freemote Emulator that permit to understand the behavior of a routing algorithm. These features are complementary as they provide three distinct levels of expressiveness. Moreover, these features can be combined in a sequential manner to understand tricky implementations. Indeed, the less expressive feature will quickly point the finger at the problem without providing a clear understanding and speed up the lookup of the same problem with the more expressive feature.

5 Conclusion and Future Work

Software development in the field of wireless sensors networks is evolving towards easier programming models by adopting widely used, general purpose languages such as Java. We developed the Freemote Emulator, a highly heterogeneous and visual emulator that aims at providing a general tool for the emerging Java based Motes. The main features of this emulator provide a useful visualization tool for understanding the behavior of a WSN at different levels from simple application flows to more tricky routing algorithm concerns. We think that this tool will help developers to quickly produce applications in this field as the Freemote Emulator provides an easy to learn programming environment and assures that the code that runs in emulation mode will also run on real nodes (although currently limited to the JMote nodes). The emulator does not focus on timing aspects but gives accurate results for evaluations that do not depend on it. In [16] this emulator has been used by our research group to evaluate the performances in terms of number of messages exchanged for a DHT based lookup algorithm described in [17] over networks ranging from 100 to 10'000 nodes.

As future work, we want to add some features to the emulator. First, we are going to improve the low layer simulation by implementing a medium access control protocol in order to increase the credibility of the results from the experiments. Moreover we are going to extend the number of Java based Motes supported by the platform such as the Sun SPOT and evaluate the adaptability with the emerging Sentilla Point Java virtual machine. Finally, we want to extend the number of bridges in an experiment in order to provide more complex topologies involving both, real and emulated nodes.

The documentation, the source code and a web startable distribution with predefined scenarios are available at `http://mote.tic.eia-fr.ch`.

References

1. Johnson, D., Stack, T., Fish, R., Flickinger, D.M., Stoller, L., Ricci, R., Lepreau, J.: Mobile Emulab: A Robotic Wireless and Sensor Network Testbed. In: Proceedings of INFOCOM, pp. 1–12 (2006)
2. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-Level Sensor Network Simulation with COOJA. In: Proceedings of SenseApp, pp. 641–648 (2006)
3. Apache Software Foundation, Apache Logging Services Project - Apache log4j, last accessed January 3th, 2008, `http://logging.apache.org/log4j/`
4. Koshy, J., Pandey, R.: Vm*: Synthesizing scalable runtime environments for sensor networks. In: Proceedings of SenSys (2005)

5. Polley, J., Blazakis, D., McGee, J., Rusk, D., Baras, J.S.: ATEMU: a fine-grained sensor network simulator. In: Proceedings of SECON, pp. 145–152 (2004)
6. Titzer, B.L., Lee, D.K., Palsberg, J.: Avrora: scalable sensor network simulation with precise timing. In: Proceedings of IPSN, pp. 477–482 (2005)
7. Eriksson, J., Dunkels, A., Finne, N., Österlind, F., Voigt, T.: Mspsim - an extensible simulator for msp430-equipped sensor boards. In: Proceedings of EWSN (2007)
8. Sundresh, S., Wooyoung, K., Agha, G.: SENS: A Sensor, Environment and Network Simulator. In: Proceedings of the Simulation Symposium, pp. 221–228 (2004)
9. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In: Proceedings of Sensys (2003)
10. Girod, L., Ramanathan, N., Elson, J., Stathopoulos, T., Lukac, M., Estrin, D.: Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks. TOSN 3 (2007)
11. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab: a wireless sensor network testbed. In: Processing of IPSN, pp. 483–488 (2005)
12. Müller, R., Alonso, G., Kossmann, D.: A Virtual Machine for Sensor Networks. In: Proceedings of EuroSys (2007)
13. Levis, P., Culler, D.: Maté: a tiny virtual machine for sensor networks. In: Proceedings of ASPLOS, pp. 85–95 (2002)
14. Levis, P., Gay, D., Culler, D.: Active sensor networks. In: Proceedings of NSDI, pp. 343–356 (2005)
15. Shaylor, N., Simon, D.N., Bush, W.R.: A java virtual machine architecture for very small devices. In: Proceedings of SIGPLAN, pp. 34–41 (2003)
16. Maret, T., Kropf, P., Hirsbrunner, B.: Un environnement d'émulation hybride pour un algorithme de DHT adapté au contexte des réseaux ad hoc. Master Thesis. Universities of Fribourg & Neuchâtel (2007)
17. Kummer, R., Kropf, P., Felber, P.: Distributed Lookup in Structured Peer-to-Peer Ad-Hoc Networks. In: Proceedings of DOA (2006)
18. Göktürk, E.: A Stance on Emulation and Testbeds, and A Survey of Network Emulators and Testbeds. In: Proceedings of ECMS (2007)
19. Rieder, M., Joye, P., Clerc, A., Maret, T., Steiner, R., Sterren, T.: ADS Project. Technical Report, HES-SO RCSO.TIC (2007)
20. Clerc, A., Joye, P., Rieder, M., Schroeter, N.: BlueBee: A ZigBee and Bluetooth extension board for PDA Java. Technical report of ADS Project, University of Applied Science of Fribourg (2007)
21. Maret, T., Martenet, N., Joye, P., Schroeter, N.: PDA Java: A Java Based Embedded System. Technical report, University of Applied Science of Fribourg (2005)
22. Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. In: Proceedings of WMCSA, pp. 90–100 (1999)
23. Gomez, C., Salvatella, P., Alonso, O., Paradells, J.: Adapting AODV for IEEE 802.15.4 mesh sensor networks: theoretical discussion and performance evaluation in a real environment. In: WoWMoM, pp. 159–170 (2006)
24. Sun Microsystems, SunSpotWorld - Home of Project Sun SPOT, last accessed (January 2, 2008), http://www.sunspotworld.com/
25. Sentilla Corporation, Sentilla — Software Architecture, last accessed (January 3, 2008), http://www.sentilla.com/architecture.html

An Efficient IP Lookup Architecture with Fast Update Using Single-Match TCAMs[*]

Jinsoo Kim and Junghwan Kim[**]

Department of Computer Science, Konkuk University,
322 Danwol-dong, Chungju-si, Chungbuk 380-701, Korea
{jinsoo,jhkim}@kku.ac.kr

Abstract. The increasing demand for new multimedia services requires the higher performance routers. The performance of Internet router highly depends on the efficiency of update operations as well as lookup operations on IP forwarding table. While IP lookup schemes based on TCAM(Ternary Content Addressable Memory) can achieve high speed lookup, they usually need more complex update operations because of the ordering constraint on prefixes. In this paper we propose an efficient IP lookup architecture to provide fast update using a new type of TCAM named single-match TCAM. Also, we present elaborated algorithms to guarantee that each single-match TCAM generates at most one match for a given destination IP address. In our scheme the updating overhead can be reduced because there is no ordering constraint on the single-match TCAM. We evaluate as well the update performance of our scheme through simulation under real forwarding tables and update data.

Keywords: Internet router, single-match TCAM, IP lookup, fast update.

1 Introduction

The qualities of wireless/mobile services as well as wired services highly rely on the performance of the Internet. A diversity of multimedia applications including mobile services has been explosively invented and the number of hosts and users has increased on the Internet. Accordingly, Internet traffic has exponentially increased as well. To guarantee the service qualities under the growing Internet traffic, it is necessary to improve remarkably the performance of the router which is a key element in the Internet.

IP address lookup is one of the most important functions in Internet routers. In order that a router forwards an incoming packet to its final destination, the router must determine the output port or the next hop address by looking up the matching prefix in the forwarding table based on the destination address of the packet. The IP lookup operation becomes more and more computationally intensive because the variable-sized prefixes have been extensively employed since the advent of CIDR(Classless Inter-Domain Routing). Since several prefixes can be matched for a destination IP

[*] This work was supported by Konkuk University.
[**] Corresponding author.

address under the CIDR, the router has the burden to select the longest matching prefix(LMP) as the best matching one. Therefore, the performance of the router strongly depends on the efficiency of the IP lookup operation.

Many researchers have studied fast lookup schemes for the development of the high performance routers[1, 2]. Most of the schemes can be classified into software approaches based on trie and hardware approaches based on TCAM(Ternary Content Addressable Memory). Trie-based IP lookup schemes usually require several memory accesses per lookup and those accesses may be serialized. In contrast, TCAM can perform a lookup operation in a single cycle owing to its parallel access characteristics. Therefore, TCAM have been paid much attention to in recent years.

TCAM is a fully associative memory in which each memory cell can store a “don’t care” state in addition to 0’s and 1’s states. Thus, TCAM can look up variable-length prefixes for a given destination address. Because there may be several matches in an IP lookup operation, it is required to determine the best match, *i.e.*, LMP. For the determination of the LMP, all prefixes of a TCAM needs to be ordered by some criteria such as length, the ancestor-descendent relationship and level on the prefix search trie. Under the ordered circumstance a priority encoder can select the LMP on the uppermost location among all matched prefixes.

The forwarding table in a router is frequently updated to avoid Internet instability[3]. In particular, Internet backbone router should be able to deal with a few hundred or thousand updates per second. Most of TCAM-based lookup schemes may experience several movements of prefix entries for a single update because the ordering must be maintained in the TCAMs. Therefore, frequent updates may consume many computation cycles in the IP lookup engine and result in the degradation of the lookup performance. The efficient update is one of the most important issues together with lookup performance and power management in TCAM-based schemes.

In this paper, we present a new architecture to provide fast update by using single-match TCAMs. Our elaborated algorithms guarantee that each single-match TCAM generates at most one match for a given destination address. So, it can eliminate both the ordering constraint and the priority encoder in a single-match TCAM, which makes the update fast. The rest of this paper is organized as follows. Related works on fast update of TCAM are described in section 2. In section 3 we propose our IP lookup architecture for fast update and describe the functionality of each component. In section 4 we present the algorithms for IP lookup, insertion and deletion of a prefix respectively. The performance of the proposed scheme is evaluated in section 5. Finally, we conclude this paper in section 6.

2 Related Works

Several methods have been proposed to reduce the updating overhead. Shah and Gupta[4] proposed the two fast updating algorithms which are PLO_OPT and CAO_OPT, to reduce the number of memory movements during update. In PLO_OPT all the existing prefixes are sorted by their lengths and free locations are reserved in the middle of the table. Then the number of memory movements per update is no more than $L/2$ where L is the maximum prefix length, *i.e.*, 32 in IPv4. CAO_OPT exploits the fact that the ordering needs to be maintained only between two prefixes one of which is the prefix of the other. In this algorithm the ordering is referred to as the chain-ancestor ordering(CAO) where a chain means the collection of the prefixes on the path from the root to a leaf node in a prefix search trie. In CAO_OPT the worst case number of memory movements per update has been reduced to *D*/2 where *D* is the maximum length of chains.

Wu et al.[5] presented an update algorithm based on the prefix level. A forwarding table is divided into several partitions according to the levels in a prefix search trie and the partitions are ordered by its level. Each partition includes free space for future update. A new prefix can be easily inserted if its parent and children are in different level partition. However, the free space may contain prefixes of different levels within the free space as update proceeds by means of that algorithm. It will cause the memory movements as in the CAO_OPT. In case of deletion it leaves the deleted space as unavailable, so it wastes memory severely. Moreover, it is difficult to predetermine the size of each free space because the distribution of updates cannot be estimated in advance.

Several approaches have been researched to remove both the ordering constraint and the priority encoder module. Kobayashi et al.[6] modified TCAM by adding vertical OR circuits in the mask column to directly select the longest mask among the matched entries without priority encoder logic. It does not require any ordering constraint, so fast update can be achieved. Actual lookup delay may be increased due to the vertical ORing, even though the delay time can be reduced by pipelining technique. Ng and Lee[7] partitioned a forwarding table into several TCAM modules so that each module only contains prefixes with the same output port number. A new prefix can be inserted at any location within a TCAM module without considering the ordering. However, many updates just change the output port numbers of the existing prefixes. Such updates lead to excessive memory movements because the prefix must be moved to another module associated with new output port.

3 Proposed IP Lookup Architecture

3.1 Conventional TCAM-Based Architecture

Conventional TCAM-based IP lookup architecture consists of a TCAM, a conventional data memory and a priority encoder as shown in Fig. 1. For a given destination IP address, there are possibly multiple matches in the TCAM and the priority encoder selects one final matched entry among those matches. The entry in the data memory which corresponds to the final matched entry of the TCAM contains target output port number. Using the output port number the packet can be delivered to the target port.

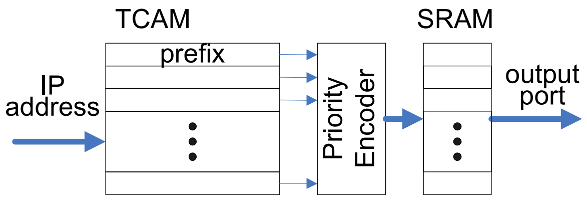


Fig. 1. Conventional TCAM-based IP Lookup Architecture

Since the priority encoder selects the final match among several matches by means of their location, the prefixes in the TCAM should be ordered so that the longest prefix always locates prior to the other matched prefixes.

For example, let's consider the sample forwarding table with 10 prefixes as shown in Fig. 2. Those prefixes can be stored in partial order of p_1, $\{p_2, p_3\}$, $\{p_4, p_5, p_6\}$, $\{p_7, p_8, p_9\}$, p_{10}, in which prefix entries within a brace can be located in any order. If an 8-bit destination address 10100100 is given, then two prefixes p_1=10100* and p_8=10* are matched. Because the prefix p_1 is located prior to the prefix p_8 in the TCAM, the priority encoder can select the prefix p_1 as the LMP. The output port 10 of the corresponding entry in SRAM can be found.

Entry	Prefix	Length	Port	Entry	Prefix	Length	Port
p_1	10100*	5	10	P_6	110*	3	14
p_2	1011*	4	11	P_7	00*	2	15
p_3	1110*	4	11	P_8	10*	2	16
P_4	010*	3	13	P_9	11*	2	17
P_5	100*	3	14	p_{10}	0*	1	18

Fig. 2. A Simple Example of Forwarding Table

3.2 Design of the Proposed Architecture

The maximum number of matched entries in a TCAM depends on the maximum depth of levels of the prefix search trie. For a given destination IP address, the prefixes which have ancestor-descendant relation will be matched simultaneously. The depth of a prefix search trie currently does not exceed 7 even including the default prefix so there can be at most 7 matches. If the forwarding table is partitioned into several TCAMs so that there is no ancestor-descendant relation in each partitioned TCAM, then it is guaranteed that there exists at most one match in each TCAM. It means that the TCAMs do not need a priority encoder any more. In section 4 we will describe how to satisfy such single-match condition when prefixes are inserted to TCAMs.

Fig. 3 shows our proposed architecture for IP lookup. It consists of 8 partitioned TCAMs and selection logic. Each of TCAMs has supplementary SRAMs which contain the lengths of prefixes and output port numbers. Note that TCAM$_0$ to TCAM$_6$ don't have any priority encoder logic whereas the last TCAM$_7$ has the priority encoder. The TCAM$_7$ is similar to the TCAM with a priority encoder used in conventional IP lookup architecture.

3.3 Single-Match TCAMs

In order to describe our single-match TCAMs, we define the terminologies which are *disjoint* and *disjoint set* as follows. Those are similarly defined in several literatures including [8].

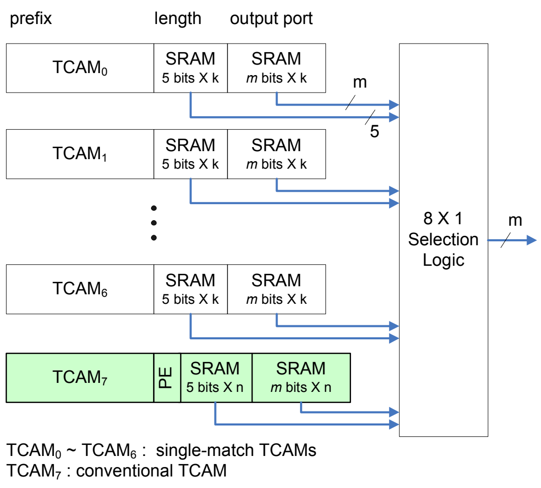


Fig. 3. Proposed IP Lookup Architecture

Definition 1. Two prefixes p_i and p_j are *disjoint* if neither p_i is the prefix of p_j nor p_j is the prefix of p_i.

For example, p_1=10100* and p_2 = 1011* of Fig. 2 are disjoint because neither p_2 can be a prefix of p_1, nor p_1 can be a prefix of p_2. However, p_8 = 10* and p_2 are not disjoint because p_8 is a prefix of p_2.

Definition 2. A set of prefixes P is the *disjoint set* if any two $p_i, p_j \in P$ are disjoint.

Each of TCAM$_0$ to TCAM$_6$ should contain a disjoint set of prefixes, *i.e.*, any prefixes in each TCAM are disjoint each other. So the result of lookup for a given IP address will be no more than one match in each TCAM. We call such TCAMs "*single-match*" TCAMs. Obviously, the single-match TCAMs don't have priority encoder logic.

The prefixes in a forwarding table are partitioned into several disjoint sets and each set is mapped onto a single-match TCAM. There is at most one match in each TCAM. Although our architecture has a conventional TCAM, the conventional TCAM will generate one final match through the priority encoder. The selection logic selects longest one among those matches by using length data. The selection logic finally sends out the corresponding output port number.

Since any prefixes which are in ancestor-descendant relation in a prefix search trie are certainly not disjoint, the depth of a prefix search trie is the minimum number of disjoint sets which is sufficient to be the number of single-match TCAMs needed. However, we need a conventional TCAM additionally in our architecture except the single-match TCAMs. The rationale for the conventional TCAM is related to the fact that the disjoint sets will be varied through several updates and it is hard to re-map those sets into single-match TCAMs by online algorithm without burden. Each single-match TCAM may be required to move the existing prefixes to another single-match TCAM in order to maintain a disjoint set when it inserts a new prefix. For example, suppose that there are only two single-match TCAMs and two disjoint prefixes

p_1=10100* and p_2 = 1011* are stored in different single-match TCAMs, then there is no way to insert a new prefix, p_8=10* into any of the single-match TCAMs without moving an existing prefix. Movements of existing prefixes are not desirable for fast updating, so we resolve the problem by providing an additional TCAM which is a conventional TCAM with a priority encoder. In case that there is no suitable single-match TCAM for a new inserting prefix, the conventional TCAM will be assigned. Since the conventional TCAM has a priority encoder, any new prefix can be inserted regardless of whether it is disjoint with the existing prefixes.

While the prefixes of the conventional TCAM need to be ordered, the prefixes of the single-match TCAMs do not. So, both insertion and deletion can be performed faster in the single-match TCAMs than in the conventional one. In section 5, the experiment result shows that most of prefixes are stored in the single-match TCAMs and very small amount of prefixes are in the conventional TCAMs.

4 IP Lookup and Update Algorithms

In this section, we describe an IP lookup algorithm to search for the LMP and also present update algorithms for inserting a new prefix into an appropriate TCAM and deleting a prefix.

4.1 Search Algorithm

Fig. 4 shows the algorithm to search for the LMP with a destination IP address, *ip_addr* as a key. Each TCAM independently performs line 2. In line 2, at most one matching prefix is found using the function match(TCAM$_i$, *ip_addr*). In case that there is no matching prefix in TCAM$_i$, *entry*[*i*].*length* becomes 0. Line 4 is performed by the selection logic. After it selects the TCAM containing the LMP, the corresponding output port number will be returned (line 5).

```
Search(ip_addr: an ip address)
1.   for i ← 0 to 7 do in parallel
2.      entry[i] ← match(TCAM_i, ip_addr)
3.   endfor
4.   Find k such that entry[k].length is the largest one
            among  entry[i].length for all 0≤i≤7
5.   return entry[k].output_port
```

Fig. 4. Search Algorithm for LMP

For example, the prefixes of Fig. 2 can constitute disjoint sets for TCAM$_0$ to TCAM$_6$ which are {p_1, p_7}, {p_2, p_4}, {p_3}, {p_5}, {p_6}, {p_{10}, p_8}, {p_9}, respectively. If an 8-bit destination address 10100100 is given, then only TCAM$_0$ and TCAM$_5$ contain the matched prefixes p_1=10100* and p_8=10*, respectively. So, *entry*[0] and *entry*[5] are set by the location of p_1 in TCAM$_0$ and that of p_8 in TCAM$_5$, respectively. Unless conventional TCAM$_7$ contains the matching prefix whose length is longer than 5 of p_1's length, the algorithm in Fig. 4 returns the value 10 of *entry*[0].*output_port*

4.2 Insertion Algorithm

Fig. 5 describes an algorithm to insert a new prefix *p* to a forwarding table. As shown in lines 1 to 6, it finds out all available single-match TCAMs to which the new prefix can be inserted. Such a TCAM satisfies that the new prefix *p* and every prefix *ep* in the TCAM should be disjoint and there should be at least one free slot in the TCAM (lines 3).

```
Insert(p: a prefix)
1.   for i ← 0 to 6 do in parallel
2.      Initialize available[i] ← false
3.      if ((p and ep are disjoint, ∀prefix ep∈TCAM_i)&&
               (there is some free space in TCAM_i))
4.         available[i] ← true
5.      endif
6.   endfor
7.   if (available[i] = false, ∀i 0≤i≤6)
8.      insert_to_tcam(p, TCAM_7)
9.   else
10.     k is randomly selected from available[i]
11.     insert_to_stcam(p, TCAM_k)
12.  endif
```

Fig. 5. Insertion Algorithm

For example, assume that the new prefix 101* is inserted under the same conditions as the example of section 4.1. Then, the prefix 101* isn't disjoint with p_1=10100*, p_2=1011* and p_8=10* which are in $TCAM_0$, $TCAM_1$, and $TCAM_5$, respectively. Consequently, *available*[*0*], *available*[*1*] and *available*[*5*] keep the *false* value. On the other hand, every prefix in $TCAM_2$, $TCAM_3$, $TCAM_4$ and $TCAM_6$ is disjoint with the new prefix 101*. Therefore, one of these TCAMs is randomly selected for insertion, if it has free space. As another example, assuming the new prefix is 1*, the prefix 1* is not disjoint with at least one prefix in each of $TCAM_0$, to $TCAM_6$. and the prefix should be inserted in $TCAM_7$.

It is easily determined if there is any non-disjoint prefix in a TCAM with respect to the new prefix. Given a 32-bit IP address TCAM searches for matched prefix in a normal lookup operation. We can give the TCAM a prefix as a search key instead of a full 32-bit IP address. If there is any non-disjoint prefix in the TCAM with respect to the prefix, match will occur in that operation. There is a simple technique to regard a prefix as a search key by considering the remaining bits of the prefix in 32-bit representation as don't care conditions[8].

If there is no available single-match TCAM in line 7 of Fig. 5, the new prefix must be inserted into $TCAM_7$ which is the conventional TCAM. Otherwise, the new prefix can be inserted into a TCAM randomly chosen from the available single-match TCAMs. The function insert_to_tcam() inserts the prefix *p* to the conventional TCAM with satisfying the ordering constraint. That functionality can be implemented by applying one of various update algorithms for the conventional TCAM. The insert_to_stcam() can simply inserts the prefix *p* into any free location in a single-match TCAM irrespective of the ordering.

4.3 Deletion Algorithm

The algorithm to delete a prefix *p* from the forwarding table is shown in Fig. 6. For the prefix deletion it is needed to determine which TCAM contains the prefix *p* as shown in line 1. Then the prefix can be deleted from the TCAM (line 2). Actually the both steps (lines 1 and 2) can be performed together. The function delete_from() is performed differently whether it operates on conventional TCAM or single-match TCAM.

We manage contiguous free space for single-match TCAM, which causes one memory movement on deletion. However, it does not need any other memory movements because there is no ordering constraint on single-match TCAM. In case of conventional TCAM the number of memory movements differs according to update algorithms. Assuming free space is contiguous in the conventional TCAM, it also requires at least one memory movement on deletion.

```
Delete($p$: a prefix)
1.  Find $k$ such that $p \in$ TCAM$_k$
2.  delete_from($p$, TCAM$_k$)
```

Fig. 6. Deletion Algorithm

5 Performance Evaluation

5.1 Simulation Environment

In our simulation we used routing tables from Route Views[9]. The update data streams for 4 weeks were used for the experiment where the data of each week were separately taken from different months. For experiment we needed to convert the routing tables into forwarding tables and filter the update data streams for the forwarding tables. Table 1 shows statistics on the forwarding table and the update data streams.

Table 1. Statistics of Sampling Data

	Jul 2007	Aug 2007	Sep 2007	Oct 2007
No. of Prefixes	243511	244095	242635	248389
No. of Updates	164467	527204	787944	651771

The number of updates only includes the number of insertions and deletions but not that of modifications of output port number. We evaluate the updating performance just by the number of memory movements incurred by updates, but the modification of the output port does not cause the memory movements.

Table 2 shows the number of memory movements per update in various updating schemes. Since there is no ordering constraint on single-match TCAM, the actual number of memory movements incurred by each update may be 0. However, we adopted a policy that free space of TCAM should be contiguous, so any deleted space needs to be compacted to the contiguous free space. It causes almost one memory movement per deletion on the average. In Table 2 the column represented by sTCAM shows the number of memory movements in single-match TCAM. The other columns show those for various updating schemes in conventional TCAM, which are summarized in [4].

Table 2. Comparison of Memory Movements

Memory Movements	sTCAM	TCAM		
		L-algorithm	PLO_OPT	CAO_OPT
Insertion	0	7.27	4.1	1.02
Deletion	1			

5.2 Simulation Results

Table 3 shows the average number of memory movements per update in our scheme. Each prefix in the forwarding table is randomly assigned to a single-match TCAM unless that prefix is not disjoint with any prefix in the TCAM. If an update occurs in single-match TCAM and the update is deletion, then the number of memory movement is calculated as one. But, if the update is insertion, there is no memory movement in single-match TCAM. In case of conventional TCAM, CAO_OPT was applied to evaluating the number of memory movements. Note that the effective update cost is cheaper in single-match TCAM than in conventional TCAM. In our scheme the average number of memory movements per update ranges from 0.4902 to 0.5061, which is half as large as CAO_OPT.

Table 3. Memory Movements per Update

Mem. Movements	Jul 2007	Aug 2007	Sep 2007	Oct 2007
Moves/Update	0.4902	0.5061	0.4954	0.4965

Fig. 7 shows the number of prefixes initially contained in each TCAM. The single-match TCAM is randomly chosen, so the prefixes can be evenly distributed among the single-match TCAMs. The number of prefixes in conventional TCAM is very small and merely 821, which is 0.33% of total prefixes. It implies that the conventional TCAM which requires the ordering constraint can be constructed as very small size in our architecture.

The good updating performance is due to the fact that most of updates are performed in single-match TCAMs and very few updates are performed in conventional TCAM. Fig. 8 shows the distribution of updates over TCAMs. Most of updates concentrate on single-match TCAMs and are evenly distributed among the single-match

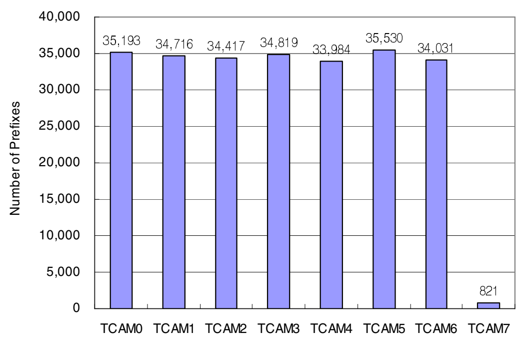


Fig. 7. The Number of Initial Prefixes in Each TCAM

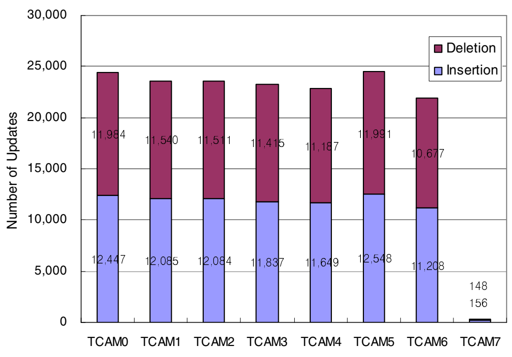


Fig. 8. Insertions and Deletions

TCAMs as well. The numbers of insertions and deletions in conventional TCAM are merely 156 and 148, which correspond to 0.19% and 0.18% of total insertions and deletions respectively. The insertion ratio, 0.19%, is much smaller than the ratio of the number of initial prefixes, 0.33%, in the conventional TCAM, which implies the relative portion of the conventional TCAM will not increase.

5.3 Discussion

The updating performance is related to two factors: the number of updates in the conventional TCAM and the number of deletions in the single-match TCAMs. In case of single-match TCAM the memory movements are only incurred by deletions while both insertions and deletions incur memory movements in the conventional TCAM.

The smaller the number of updates in the conventional TCAM is, the better updating performance is achieved. The simulation results show that the number of updates in the conventional TCAM is quite small. In the simulation a simple random assignment strategy was applied to the insertion algorithm, however, it is possible to apply various assignment strategies to the insertion algorithm. It is expected that the number of prefixes and updates in the conventional TCAM will be affected by the assignment strategies.

The number of single-match TCAMs may be controlled depending on cost and performance. It is also expected that even if the number of single-match TCAMs is reduced, the performance will not be rapidly degraded because large amount of prefixes can reside in small number of disjoint sets.

6 Conclusion

The conventional TCAM must satisfy some ordering constraint on prefixes for longest prefix matching. It results in low updating performance of the TCAM and even lookup performance would be low due to overhead of a priority encoder.

Our novel architecture shows good performance in updating by introducing a new type of TCAMs so called single-match TCAMs which do not need any priority encoder logic. By using an elaborated assignment strategy to insert a new prefix, each single-match TCAM can maintain a disjoint set of prefixes and produce at most one match. Since it does not require any ordering constraint on each single-match TCAM, a newly inserted prefix can be located at any place within a single-match TCAM. It is expected that lookup will also spend less time than in the conventional TCAM because it does not have a priority encoder.

Although the proposed architecture still requires a conventional TCAM, our simulation result shows that the number of prefixes which reside in the conventional TCAM is very small. As the result very few updates are performed in the conventional TCAM and the average number of memory movements per update is as small as about 0.5.

Novel assignment strategies for prefix insertion should be developed and evaluated in further research. The memory movements on deletions in single-match TCAM can be eliminated provided that the insertion hardware is enhanced. The design of the hardware to eliminate memory movements also remains for future work.

References

1. Ruiz-Sanchez, M.A., Biersack, E.W., Dabbous, W.: Survey and Taxonomy of IP Address Lookup Algorithms. IEEE Network 15, 8–23 (2001)
2. Chao, H.J., Liu, B.: High Performance Switches and Routers. Wiley-Interscience, Chichester (2007)
3. Labovitz, C., Malan, G.R., Jahanian, F.: Internet Routing Instability. IEEE/ACM TON 6, 515–528 (1998)
4. Shah, D., Gupta, P.: Fast Updating Algorithms for TCAMs. IEEE Micro 21, 36–47 (2001)
5. Wu, W., Shi, B., Wang, F.: Efficient location of free spaces in TCAM to improve router per-formance. IJCS 18, 363–371 (2005)
6. Kobayashi, M., Murase, T., Kuriyama, A.: A Longest Prefix Match Search Engine for Multi-Gigabit IP Processing. In: 2000 International Conf. on Communications, pp. 1360–1364. IEEE Press, New Orleans (2000)
7. Ng, E., Lee, G.: Eliminating Sorting in IP Lookup Devices using Partitioned Table. In: 16th IEEE International Conf. on Application-Specific Systems, Architecture and Processors (ASAP), pp. 119–126. IEEE Press, Greece (2005)
8. Akhbarizadeh, M.J., Nourani, M., Cantrell, C.D.: Prefix Segregation Scheme for a TCAM-Based IP Forwarding Engine. IEEE Micro 25, 48–63 (2005)
9. University of Oregon Route Views Project, `http://www.routeviews.org/`

OMEN – A New Paradigm for Optimal Network Mobility

Pedro Vale Pinheiro[1] and Fernando Boavida[2]

[1] Centro de Informática
vapi@ci.uc.pt
[2] Departamento de Engenharia Informática,
Universidade de Coimbra, 3030-290 Coimbra, Portugal
boavida@dei.uc.pt

Abstract. Current mobility solutions are anachronistic. They assume that nodes as well as networks must not be aware of mobility and behave as if they were static, and that any new developments must preserve that assumption. This is putting unnecessary pressure on network devices – such as routers and mobile routers – leading to a complex and under-performing network, contradicting the fundamental paradigm of the Internet: keep the network as simple as possible. In the current paper a solution for optimized network mobility that assumes nodes and networks are mobility-aware is presented. The solution is analysed and compared with two other – NEMO Basic Support and MIRON – showing that its advantages are clear.

Keywords: Network mobility, route optimization, nested mobile networks.

1 Introduction

Node mobility has been the subject of considerable study for more than a decade. Although to a lesser extent, due to the fact that the need for it is still gaining momentum, network mobility (NEMO) has also been the subject of research in the last few years. In spite of this, we can still say that there is a long way to go before efficient and effective node and network mobility solutions are in place and largely deployed.

The problems that arise from moving an entire network, possibly containing other mobile networks inside it, in what is known as a nested scenario, are not trivial. The NEMO Basic Support Protocol, RFC 3963 [1], has been developed in order to provide a basic solution for network mobility, having as main requirement complete mobility transparency, that is, requiring no need for any modifications to the nodes. However, the positive aspects of RFC 3963 – its simplicity and its lack of requirements on the nodes – are at the very basis of its weaknesses. As is, RFC 3963 has several limitations that urge to be solved and it can only be looked at as an acceptable solution that provides network mobility with minimum impact, at a high cost in terms of efficiency and performance.

The main problem of existing proposals for network mobility – RFC 3963 included – is the fact that all of them assume that minimal impact, or even no impact, on the nodes is an absolute must. The immediate consequence of this is that the proposed solutions impact the network, as opposed to the nodes, which is far more negative and, in fact, contradicts the basic Internet principle that states that complexity should be inside the hosts and not inside the network.

A paradigm shift is, thus, necessary, in what concerns network (and node) mobility. As more and more Internet nodes and networks become mobile, it is fundamental to make nodes and networks aware of their mobility. Not doing so is ‘to hide the head in the sand’ or, which is worse, to force the current Internet to operate and behave as the Internet of the 1970s. We are inexorably approaching an era where Internet mobility will be the rule, not the exception. In this new era, nodes must have enough ‘intelligence’ to know if they are mobile or not and to react accordingly. On the other hand, the Internet must be kept simple and the protocols – not the nodes – must be kept unchanged as far as possible.

In line with the above, it is the objective of this paper to present a new approach to network mobility that requires no modifications to the existing protocols. The proposed solution, called OMEN (Optimised Mobility for Enhanced Networking), is based on the assumption that nodes and networks are aware of their mobility condition. This simple paradigm shift enables the development of a mobility solution that is, at the same time, simple, efficient and effective. The proposal focuses on the IPv6 scenario only.

Related work is addressed in Section 2. This includes the NEMO Basic Support Protocol [1] and MIRON [10-12]. The OMEN approach is presented and discussed in Section 3. This is followed by an evaluation of the presented proposal by comparison with the two proposals mentioned in the related work section (namely, NEMO Basic Support and MIRON). This comparison, presented in Section 4, was made by simulation, using a simple simulator developed by the authors for this purpose. Section 5 summarises the key features of OMEN and identifies guidelines for further work.

2 Related Work

Although several protocols, schemes and proposals for network mobility have been developed in the past [5-9], they basically consist of variations and add-ons to the NEMO Basic Support Protocol [1]. One exception is the MIRON proposal [10-12]. In the following sections we will briefly present NEMO Basic Support and MIRON, highlighting their main characteristics and limitations. Both approaches to network mobility are constructed on the assumption that there must be no changes to the mobile network nodes (MNN) and to the correspondent nodes (CN).

2.1 NEMO Basic Support Protocol

The idea behind NEMO Basic Support is to readily allow network mobility without the need to change MNNs and CNs. All the tasks inherent to network mobility are, thus, carried out by mobile routers (MR) and home agents (HA).

Figure 1 illustrates the operation of NEMO Basic Support (in this figure, and from now on, NEMO stands for mobile network). Whenever a packet destined to the mobile network prefix (MNP) arrives at the home network, the HA encapsulates the packet and sends it to the care-of-address (CoA) of the mobile router over the MRHA tunnel. The MR will then decapsulate the packet and deliver it to the MNN. Conversely, packets from an MNN to a CN are encapsulated, sent from the MR to the HA over the MRHA tunnel, decapsulated at the HA and routed to the CN.

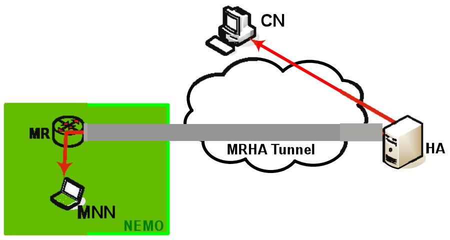


Fig. 1. NEMO Basic Support operation

Although extremely simple and fully compatible with legacy devices, NEMO Basic Support suffers from several important problems, such as triangular routing, potential bottleneck in the home network and amplified sub-optimality in nested mobile networks. These problems are thoroughly discussed in [2] [3] and [4].

Virtually all problems affecting the NEMO Basic Support approach have to do with lack of route optimization. Performing route optimization is, nevertheless, no simple task, and several issues must be taken into account when pursuing it: additional signaling overhead, extending nodes with new functionalities, increased protocol complexity, processing load, increased delay during handoff, scalability, mobility transparency, location privacy, security considerations and support of legacy nodes.

2.2 MIRON

MIRON (Mobile IPv6 Route Optimization for NEMO) [10-12] addresses the problems of NEMO Basic Support, by proposing some route optimization mechanisms. The optimization mechanisms differ, depending on the type of mobile network node.

In the case of local fixed nodes (LFN) and local mobile nodes (LMN), all traffic to/from these nodes must be optimized by the mobile router. This means that MRs must keep track of all LFN-CN optimizations and must perform route optimization, using the return routability mechanism, whenever the mobile network moves.

In the case of visiting mobile nodes (VMN), the approach is different. MIRON prescribes the use of an address delegation mechanism, based on PANA (Protocol for Carrying Authentication for Network Access) [13], which provides these nodes with topologically meaningful addresses. In this way, the MRs ask from their foreign network an IP address for each VMN, and route packets to these addresses. MRs still have to enable these addresses to be routable inside the NEMO and perform source address routing in the MR in order to send VMN's packets directly. Whenever an MR changes network it must request new IP addresses for all of its VMNs.

In MIRON, nested networks are treated as VMNs. In addition to the requirements identified above for VMNs, every MR in the nested NEMO needs to keep track of the addresses of all the nodes requesting IPv6 addresses, in order to create and maintain the respective routing entries.

3 Optimised Mobility for Enhanced Networking (OMEN)

3.1 The OMEN Paradigm

The very basis of OMEN is a change in the paradigm of mobility access control. So far, the solutions focused on the maximum mobility transparency for the nodes. OMEN proposes the opposite: nodes must be aware of their mobility condition. So, when a mobile router acquires a new care-of-address it should inform its inside network. Mobile network nodes that understand this advertisement may then use the MR's CoA as their own new CoA.

Instead of creating a new protocol for informing MNNs of their MR's CoA, OMEN uses Neighbor Discovery (RFC 4861) [15]. Thus, either as response to a router solicitation message or by its own initiative, an MR can send router advertisement messages that will be used by MNNs to learn their CoA. The CoA will be carried in a new option. As the definition of new options is already accounted for in RFC 4861, there is no need to change the protocol.

Any node intending to optimize the route for a CN should use its MR's CoA. When the MR receives an optimised packet destined to its CoA it should check if the next hop corresponds to its mobile network prefix or if it is registered in its routing table, and should route the packet accordingly. The return routability procedure can be carried out without any problem, as the CN→CoA and CN→HoA routes are still available. Moreover, the CN does not need to know if it is communicating with a node under a NEMO or with an MIPv6 node.

The OMEN approach eliminates several of the problems inherent to NEMO Basic Support and to the MIRON approach, as explained below.

In OMEN, MNNs decide when to optimize routes and can perform that optimization by themselves. Thus, MRs are not burdened with this task. On the other hand, as MNNs use the MR's CoA, there is no need for MRs to request new IP addresses for each VMN or for each nested mobile network, as in the case of MIRON. When a packet arrives at an MR, it knows how to route the packet based on the next hop field in the route optimization packet header, without the need to decapsulate the packet or terminate tunnels.

In the case of LMNs, there is no need to notify the HA while they are in their home mobile network as this is automatically done by the MR. Thus, non-optimised traffic will always use the MRHA tunnel, without the need for the LMN to perform any action. Nevertheless, optimized traffic to/from the CN flows just like in the case of MIPv6 route optimization. Thus, OMEN leads to the benefits inherent to MIPv6 without requiring a binding update (BU) for each LMN.

Last but not least, as all mobility-capable nodes create optimized routes using the MR's CoA, it is possible to determine if the CN is in the same mobile network, thus opening the possibility for the communication to take place inside the mobile network, even when there is no connectivity to the outside.

3.2 OMEN and the Different Types of Mobile Nodes

OMEN uses different approaches depending on the type of MNN, as illustrated in Figure 2.

Local fixed nodes (LFN) are not aware of mobility. OMEN was not designed for this type of nodes. We will assume they are legacy or light devices, and so they do not implement OMEN. These nodes fall into the NEMO Basic Support category and, thus, will be subject to its inherent problems and limitations.

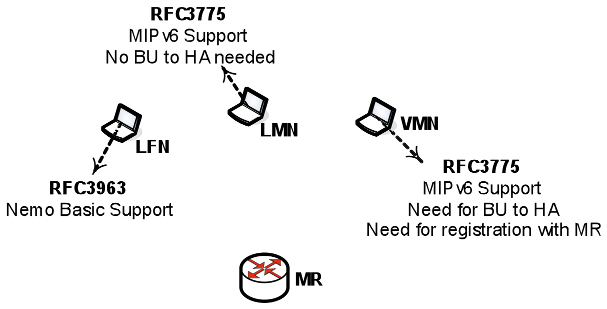


Fig. 2. OMEN and the different types of mobile nodes

A local mobile node (LMN) should accept its new CoA and should create route optimization tunnels to its correspondent nodes with its new CoA. An LMN node, while in its home network, should never inform the home agent if it acquires a new CoA, as this is the MR's job. This is so because although the mobile network is outside its home network, the LMN is inside its home (mobile) network. If the LMN leaves its home network, i.e. its NEMO network, it should follow the MIPv6 standard or act as a visiting mobile node (VMN), as explained below. In such case, the LMN should perform the return routability procedure in order to create a bi-directional tunnel with the CN.

A VMN should accept its new CoA in the same way as an LMN, and should also register its home address (HoA) with its parent mobile router, as illustrated in Figure 3. The MR must create and maintain a routing table that contains the mapping between the home addresses of its VMNs and their link local address inside the NEMO. As the next hop inside the bi-directional tunnel between CN and VMN is always the HoA, the MR can use this table to determine the link local address to which the packet should be sent. All the route optimization actions should be as specified in RFC 3775, although with the use of the new CoA. The communication between the MR and the VMN will be through the local-link address. Thus, when a VMN enters a NEMO it will immediately be informed of that fact and of which CoA to use. It should then send a binding update (BU) to its HA, notifying it of its new CoA, and register with the MR at the same time, so that when the HA answers the MR already knows to which node it must deliver the binding acknowledge packet. After the registration phase, the VMN should notify all its CNs of its new CoA.

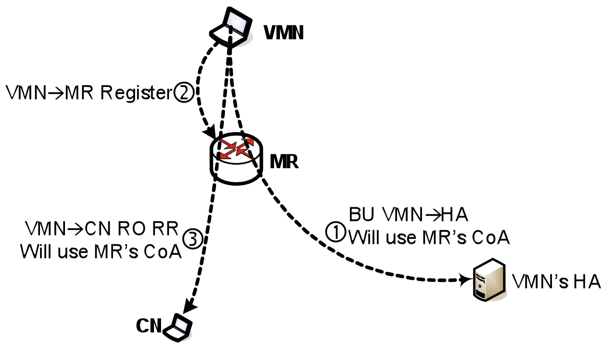


Fig. 3. VMN procedures for route optimization

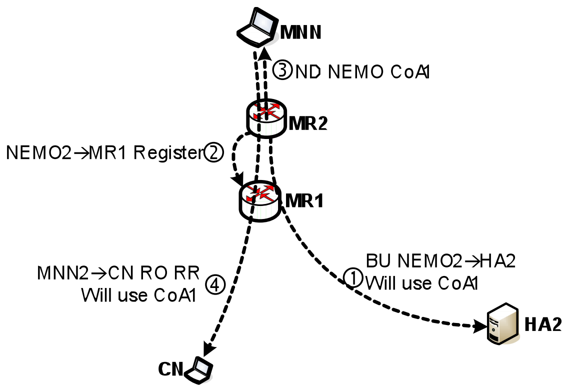


Fig. 4. Nested NEMO procedures for route optimization

Similar to the VMN case, a nested sub-MR should inform its parent MR which mobile network prefix it owns, and this information should be propagated up until the root MR, so that every mobile router can populate its routing table with this new prefix, as we can see at Figure 4. In the same way, every time a node becomes unreachable this information should be propagated upwards to all MRs until the root MR. Every MR only knows the link-local address of the sub-MR for every MNP, in the same way as a routing table.

4 Comparison with Other Approaches

In this section, comparative performance tests concerning NEMO Basic Support, MIRON and OMEN are presented. The results were obtained by simulation, using a simulator developed by the authors. Although extremely flexible and light, the simulator does not provide absolute values, its objective being to obtain relative results, suitable for comparison purposes only.

In the following sections, results of extensive testing are presented. Two different scenarios were analysed – non-nested scenario and nested scenario – each leading to its own test suite. Each test suite comprised the measurement of the round trip time from CN to MNN and back (a simple round trip time analysis) for a total of 240,000 packets. Each test suite was composed of 600 individual test runs, each comprising 400 packets (it took about 4 days to complete each test suite).

4.1 Non-nested Scenario

In this scenario, the mobile network travels from network to network getting closer to the CN's network (see 1 and 2, in Figure 5). On the last step (3) it jumps to the CN's network.

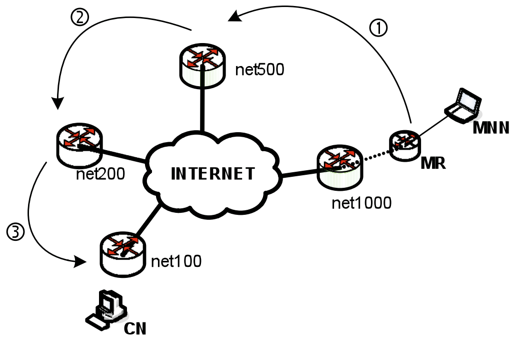


Fig. 5. NEMO movement for the non-nested scenario

The CN starts by sending 100 packets with the MR at home, and then the NEMO moves to the net500 network, where it receives 100 packets more. The same happens in the movement to net200 and net100. At net100, the MR is in the same network as the CN. The obtained round trip times are presented in Figure 6. In this figure it is possible to observe the effect of the NEMO jumps from one network to the next.

As we can see, in the case of the NEMO Basic Support, the closer the MR gets to the CN the longer it takes for the packet traverse all the networks from CN to HA to MNN and back. On the other hand, the MIRON and OMEN approaches get faster when the mobile network comes closer to the CN. In terms of performance, we can observe that MIRON and OMEN are equivalent.

For this scenario, we also analysed the handoff time, that is, the time needed to establish the connection between CN and MNN (Figure 7). For the NEMO Basic Support case this is only the time taken by the binding dialogue between MR and HA, while in the MIRON and OMEN cases this time also includes the time taken by the route optimization procedure (hence, the higher values in Figure 10).

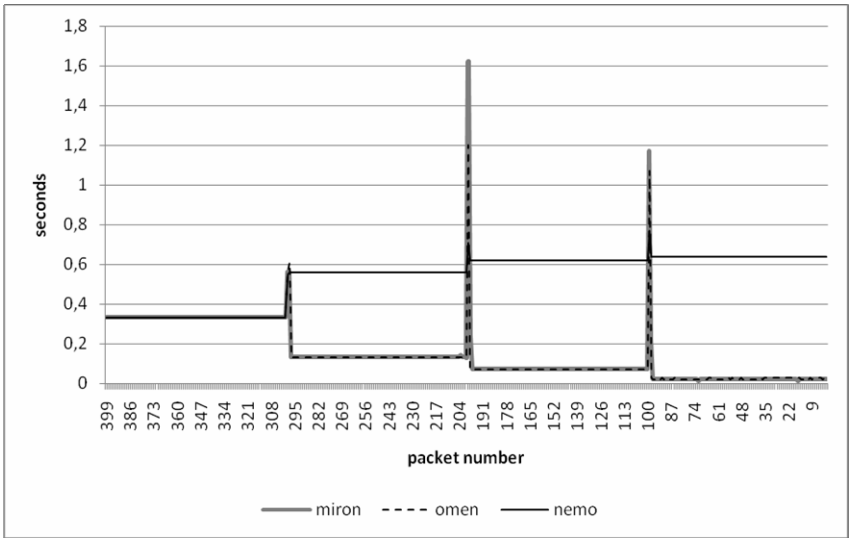


Fig. 6. Round trip time between CN and MNN (non-nested scenario)

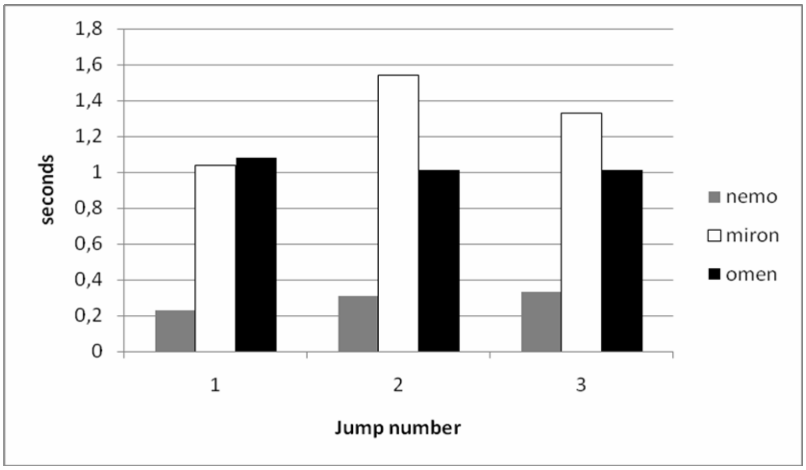


Fig. 7. Handoff time (non-nested scenario)

4.2 Nested Scenario

For the nested scenario it was only possible to compare the NEMO and OMEN approaches, as the MIRON approach revealed itself too difficult to implement. In this scenario (Figure 8), the mobile network jumps to a network near the CN and then starts getting deeper into the nested hierarchy. Each nested network has a different home network and home agent.

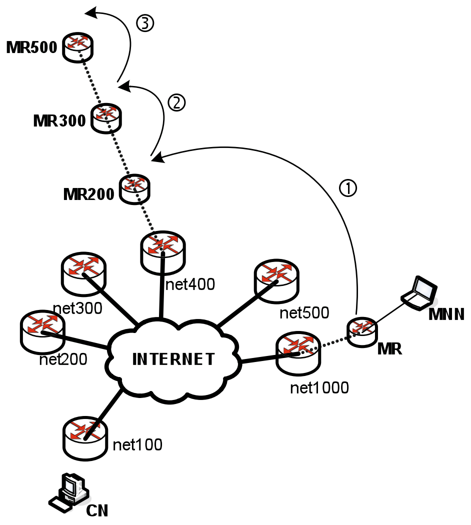


Fig. 8. NEMO movement for the nested scenario

As we can see in Figure 9, the NEMO Basic Support approach gets worse as the NEMO moves. On the other hand, OMEN can deal well with nested situation, as all MRs know the path to the MNN every time its MR moves.

Regarding the handoff time, for ease of comparison the authors opted to analyze the binding dialogue for both NEMO and OMEN on one side, and the route optimization between CN and MNN in another bar (omen-RO). The obtained values are presented in Figure 10. As we can see, the NEMO Basic Support binding update dialog gets worse as the nesting becomes deeper, while OMEN's barely changes. The same happens in terms of route optimization between CN and MNN.

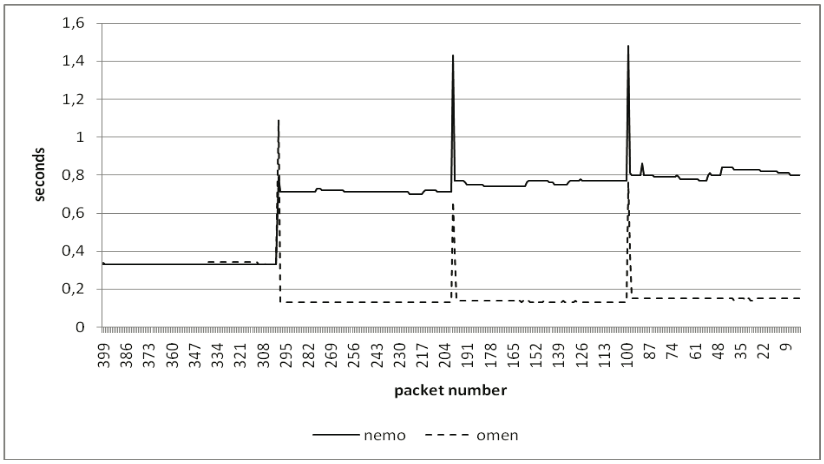


Fig. 9. Round trip time between CN and MNN (nested scenario)

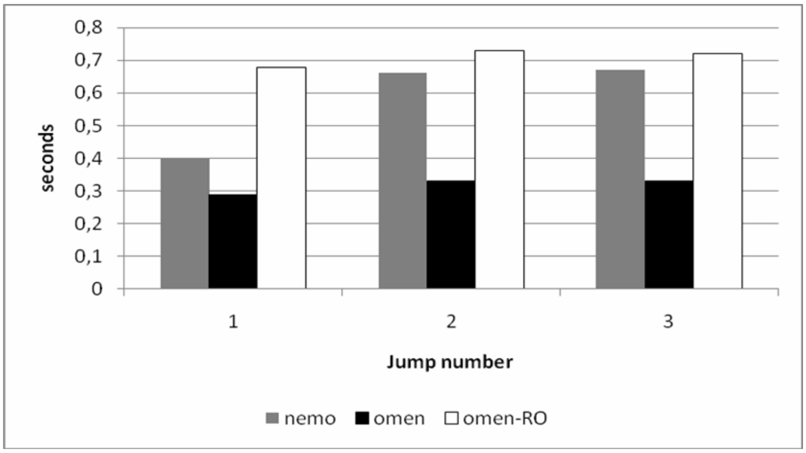


Fig. 10. Handoff time and OMEN route optimization time (nested scenario)

5 Conclusion

In this paper we presented OMEN, a solution for optimized network mobility that constitutes a paradigm shift in relation to previous approaches. In OMEN, mobile network nodes are aware of their mobility condition. OMEN has been compared with two key solutions for network mobility: NEMO Basic Support and MIRON. Considering the presented analysis and the obtained performance results, we can summarise the benefits of OMEN in the following:

1. Optimized routes are established between mobile network nodes and their correspondent nodes, and the mobile router acts as a mere routing device without specific or stringent requirements at protocol and performance levels;
2. Route optimization decisions are taken by MNNs, when and if they really need optimization;
3. Every time a mobile router acquires a new care-of address, the nodes can be immediately informed using a neighbor advertisement packet (already defined in RFC 4861, i.e., no need to develop new protocol or change existing ones);
4. As every MNN can act at its own time, the chance of having a bottleneck is lower than in the scenario where mobile router optimize routes on behalf of MNNs;
5. Visiting mobile nodes and nested NEMO networks are greatly benefited as they can create routes for correct delivery inside the network, even if there is no outside network connectivity at the moment;
6. As the optimization is on behalf of a CoA, that is, a topologically correct IP address, there is no need to traverse the home agent and so the packets are not subject to the problems of triangular routing;

Recently, [14] defined several requirements (Req) and desirable (Des) features for route optimization on a consumer electronics scenario. It is interesting to note that OMEN satisfies most of them. Req1 says that the mobile router must perform route optimisation (RO) on behalf of LFN. In this situation one must use a MIRON-like solution, as OMEN does not account for this. Req2 requests low processing load, which OMEN satisfies, as it does not require much more than forwarding packets. Req3 concerns security, which OMEN offers through MIPv6 without any change to the protocol. Des1 specifies an MR-to-MR route optimization, and as OMEN uses the CoA of the root-MR, then it gets the best possible RO result. Des2 specifies a nested-NEMO route optimization, which OMEN already complies with. Des3 specifies intra-NEMO route optimization, which OMEN also satisfies, as explained before.

In short, OMEN leads to lighter mobile routers, requires no changes to existing protocols, has better performance than the NEMO Basic Support Protocol and slightly better performance than MIRON, has less complexity than MIRON, has no limitations on the degree of nesting and, last but not least, requires no modifications to correspondent nodes or to other Internet devices with the exception of mobile routers.

Although OMEN revealed itself a good solution, considerable work still needs to be done. Future work will address prototyping and testing in environments as real as possible. Detailed consistency and robustness analysis will also be addressed. Another topic that will deserve near-future attention is the support of MIPv6 in LFNs. In this respect, the authors will analyze the feasibility of implementing OMEN support in LFNs. Last, but not least, thorough comparison between nested RO solutions and OMEN is already under way.

Acknowledgments. The work presented in this paper was partially financed by the Portuguese Foundation for Science and Technology, FCT, and by the IST FP6 CONTENT Network of Excellence (IST-FP6-0384239).

References

1. Devarapalli, V., et al.: Network Mobility (NEMO) Basic Support Protocol. RFC 3963, Internet Engineering Task Force (January 2005)
2. Ng, C.-W., et al.: Network Mobility Route Optimization Problem Statement. draft-ietf-nemo-ro-problem-statement-03, Internet Engineering Task Force (September 2006)
3. Ng, C.-W., et al.: Network Mobility Route Optimization Solution Space Analysis. draft-ietf-nemo-ro-space-analysis-03, Internet Engineering Task Force (September 2006)
4. Bernardos, C.J., et al.: NEMO: Network Mobility in IPv6. Upgrade IV(2) (April 2005)
5. Wakikawa, R., et al.: Optimized Route Cache Protocol (ORC). draft-wakikawa-nemo-orc-01, work in progress, Internet Engineering Task Force (November 2004)
6. Wakikawa, R., et al.: ORC: Optimized Route Cache Management Protocol for Network Mobility. In: 10th International Conference on Telecommunications, February 2003, vol. 2, pp. 1194–1200 (2003)
7. Na, J., et al.: Route Optimization Scheme based on Path Control Header. draft-na-nemo-path-control-header-00, work in progress, Internet Engineering Task Force (April 2004)
8. Na, J.: Seoul National University, Supporting Route Optimization in Network MObility (NEMO), Personal Wireless Communications. In: IFIP TC6 9th International Conference (September 2004)
9. Thubert, P., et al.: Global HA to HA protocol, draft-thubert-nemo-global-haha-01.txt. work in progress, Internet Engineering Task Force (October 2005)

10. Bernardos, C., et al.: MIRON: MIPv6 Route Optimization for NEMO. In: ASWN 2004 - 4th Workshop on Applications and Services in Wireless Network, Boston University, Boston MA, USA (August 2004)
11. Bernardos, C., et al.: Mobile IPv6 Route Optimisation for Network Mobility (MIRON). draft-bernardos-nemo-miron-00, work in progress, Internet Engineering Task Force (July 2005)
12. Bernardos, C.: Route Optimisation for Mobile Networks in IPv6 Heterogeneous Environments., PhD thesis, Universidad Carlos III de Madrid, Spain (September 2006)
13. Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., Yegin, A.E.: Protocol for Carrying Authentication for Network Access (PANA). draft-ietf-pana-pana-11.txt, work-in-progress, Internet Engineering Task Force (March 2006)
14. Ng, C.-W., et al.: Consumer Electronics Requirements for Network Mobility Route Optimization. draft-ng-nemo-ce-req-01, work-in-progress, Internet Engineering Task Force (November 2007)
15. Narten, T., et al.: Neighbor Discovery for IP version 6 (IPv6). RFC 4861, Internet Engineering Task Force (September 2007)

Security and Accounting Enhancements for Roaming in IMS

Seppo Heikkinen

Tampere University of Technology
P.O.BOX 553
FIN-33101 Tampere, Finland
`firstname.lastname@tut.fi`

Abstract. As the multimedia services are gaining popularity, the operators are seeking new architectures, such as IP Multimedia Subsystem (IMS), that would allow provision of these services with sufficient level of quality and security. In the future, however, it is not anymore so clear who is an operator, because the ubiquitous communication visions enables every player to interact in multitude of ways with other entities and provide services of their own. In this paper we investigate a setting, where a roaming subscriber wishes to receive service from an operator, who has no previous relationship with the home operator. We propose methods based on cryptographic identities which enable the each party to get assurance about the authenticity of each participant and the accountability of the executed actions. While suggesting completely new mechanisms for existing systems, the proposal also addresses the needs to leverage the available infrastructures in a convenient way.

Keywords: cryptographic identity, HIP, IMS, roaming, security.

1 Introduction

The operators are seeking to increase their revenues by introducing new service architectures, which would allow them to get better hold of their customers and provide them high margin value added services in an operator controlled environment. While this is understandable from the operator point of view as they battle against the flexible and innovative service providers, but the tendency towards openness and dynamic environments does not entirely support these aims. After all, in forward looking project, such as partially EU funded Ambient Networks [1], the future visions suggest that the operator landscape is bound to change. There will be a larger amount of operators, which can function in a limited scope. For instance, a single individual could assume the role of a "mini-operator" and provide access services to other nearby individuals. While this is already taking place on smaller scale, especially with WLAN networks, there are still requirements for security and compensation related issues, before this can be said to take place in larger scale.

In this paper we take a look at one service architecture, IP Multimedia Subsystem (IMS), developed within 3GPP to take into account the needs of the operators for service provisioning in an IP world. However, we base our initial assumptions on the future visions, where the operator relationships are not so clear. In other words, the multitude of different kinds of operators makes it difficult to have mutual agreements between all of them unlike today where the pre-established roaming agreements between big operators are a commonplace. This dynamic setting requires stronger means to ensure the security of the communication and the correct accounting between unknown entities as the current assumptions of IMS security rely on the good and reliable behaviour of the known operators.

For securing the initial attachment we take the approach originally developed within the Ambient Networks project, where a Host Identity Protocol (HIP) based network attachment protocol was used to connect networks and exchange configuration information. This is integrated with the IMS architecture to allow enhanced interaction that takes into account the liability and reliable identification needs of the different entities partaking in communication, i.e. the user, the access operator, and the home network. The scheme is further enhanced with hash chain based approach, which allows non-repudiative accounting records to be made. In other words, if the user has used the service, this cannot be denied afterwards. Similarly, if the user does not seem honest in its service usage in terms of compensation, the service is no longer provided. Overall, the presented scheme takes a cross-layered approach as the same identities are used to secure actions on different layers and there is a tighter interconnection between them.

This paper is organised as follows. In the next section we go through some of the HIP basics. The third section provides a brief overview of IMS functionality. In the following two sections after that we outline the workings of our architecture and the idea behind the hash chain based service usage, respectively. The sixth section provides evaluation of the architecture in terms of security properties. The seventh section gives some directions for the future work. Finally, the seventh section concludes the paper.

2 HIP Basics

In order to attach entities in our scheme, one needs a way to exchange identity and configuration information using a secure channel in a heterogeneous network environment. In Ambient Networks such attachment procedure was proposed to be developed from the work done on HIP and due to its identity based approach it is quite natural choice for this work as well.

HIP is a proposal for future network architectures that introduces a new identity layer between the network and transport layers [2]. This allows decoupling of the dual role of the IP addresses. That is, currently they function as identities and locators. In the HIP model the end points are identified by their cryptographic identities, called Host Identity Tags (HIT), which are formed from their public keys using hashing. This accommodates for end host authentication and simple key exchange. Thus, the parties are able to form a security association between themselves, which can be used to protect the control information exchange. Additionally, the protection of subsequent data exchange is possible with IPsec ESP [3]. HIP uses four messages in the so called base exchange to establish the identity of the parties and to create the needed keying material with the help of Diffie-Hellman key exchange (see Fig. 1). The procedure also takes into account the denial of service concerns by introducing a puzzle scheme, which forces the initiator to spend computing resources before the target is willing to store any state related to the communication.

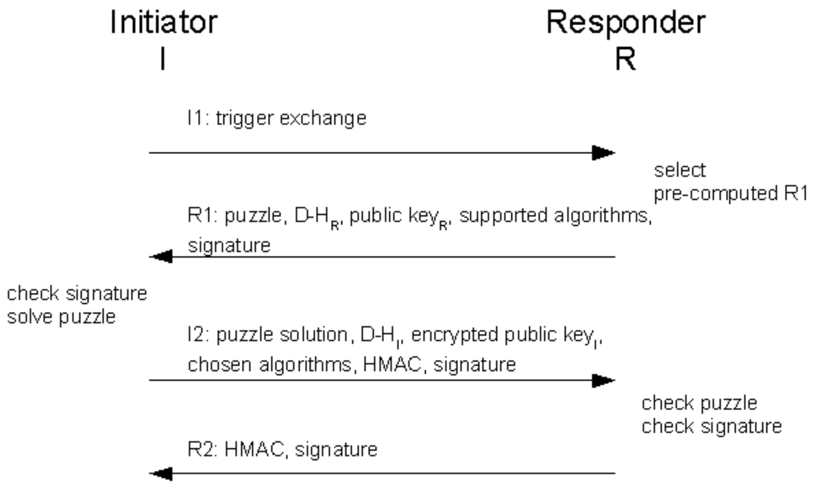

Initiator
I
Responder
R
I1: trigger exchange
select
pre-computed R1
R1: puzzle, D-H$_R$, public key$_R$, supported algorithms, signature
check signature
solve puzzle
I2: puzzle solution, D-H$_I$, encrypted public key$_I$, chosen algorithms, HMAC, signature
check puzzle
check signature
R2: HMAC, signature

Fig. 1. HIP base exchange showing the four handshake messages for association establishment

3 IMS Basics

IP Multimedia Subsystem (IMS) is an architecture designed by 3GPP to facilitate the provisioning of multimedia services over IP packet networks [4]. While IMS is mostly envisaged to be used in UMTS networks, it is basically access agnostic, so one can also use it, for instance, in WLAN and fixed line environments as well.

At the heart of IMS is Session Initiation Protocol (SIP), which is seen as a general signalling protocol running on top of transport layer, i.e. it is used for establishing and controlling the sessions between the communicating entities [5]. SIP works in a hop-by-hop fashion, so each network element, or proxy, on the path can make its own changes to the messages in order to provide either additional services or ensure the correct routing of the signaling messages. SIP is basically a simple text based protocol and the individual messages contain a header and a body section, much in the way of HTTP. Headers include most of the control information and the body section usually contains information regarding the information content the parties are negotiating about. It could be, for example, used to describe the media session to be negotiated and use a different protocol, such as Session Description Protocol (SDP). In theory, however, SIP message body could contain any other type of content as well.

The simplified architecture is depicted in Fig. 2 in terms of session establishment between two users. The more detailed architecture can be found from [4]. The first SIP contact point for the user is Proxy Call Session Control Function (P-CSCF). It is responsible for finding the next contact point, which in the case of home network might be Serving CSCF (S-CSCF) or in case P-CSCF is in the visited network, then Interrogating CSCF (I-CSCF) of the home network that the home network uses as a published entrance point to its network. P-CSCF could also use other border elements, such as Interconnection Border Control Function (IBCF), to take care of the communication with the other networks. P-CSCF can also interact with transport level entities, so it can set policies for the handling of the data traffic of the user. I-CSCF is also responsible for finding an appropriate S-CSCF in the home domain to serve the roaming user. S-CSCF is the "work horse" of IMS system as it is responsible for authenticating the user with the help Authentication and Key Agreement (AKA) procedure. It does it in cooperation with the Home Subscriber Server (HSS), which contains all the subscriber information. S-CSCF is also in the path of every SIP message the user sends or receives, so it can redirect the messages to the other networks or the appropriate application servers (AS) as dictated by the profile of the user. There are also other network elements to take care of the media processing and interaction with other networks, such as the legacy telephone systems, but in order to keep things simple in our scope they are not shown in the figure nor discussed further.

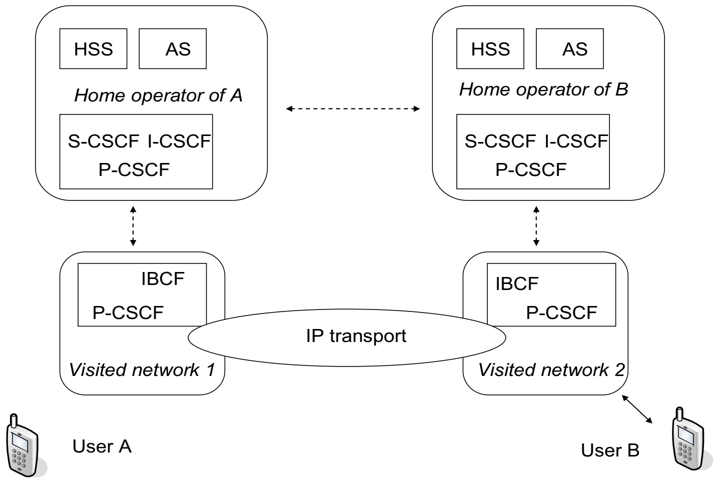


Fig. 2. Simplified IMS architecture

As mentioned above, AKA procedure is used to authenticate the user and it is based on the shared secret, which reside both in the Subscriber Identity Module (SIM) of the user and HSS of the home operator. It is basically a challenge-response protocol and it also creates keys for protecting the subsequent communication between the user and P-CSCF. This can be done, for instance, with the help of IPsec. The procedure employs similar functionality as in UMTS-AKA, but uses the SIP layer as IMS is expected to be access network agnostic. Security of the overall system is based on the assumption that the communicating operators are trustworthy and they have secured their own internal network elements [6]. From this follows that the typical SIP level identities do not provide any security of their own and the content of communication is expected to be reliable. Thus, it is assumed that the operators have pre-established security associations with themselves that usually results from creating a roaming agreement. This is heavily a manual process. While the inter-operator connection needs to be secured, it is not mandatory to secure the connections between the internal network elements of the operator.

4 System Overview

The following section describes the interaction of the involved entities on a high level. We suggest enhancements to the way the network level attachment is done between the various parties of the IMS roaming scenario. In addition to the network level modifications, SIP messages require additional fields to ensure the transporting of identity and authorisation information between the entities.

At the beginning of the procedure the user is made aware of the existence of the access network through a beacon message sent by the local access point. Any further messages are forwarded deeper into the network, similarly as is done with access point controllers [7]. In our figure (Fig. 3) this element is marked as P-CSCF, although in real implementation it could be another network level device, which only shares an interface with P-CSCF. The user initiates the attachment procedure using the HIP like network attachment protocol, as described in [8]. This includes the basic properties of HIP handshake, through which the parties are able to negotiate the session parameters and the keys. They also authenticate each others identifiers. The enhanced properties include the possibilities of providing configuration and other network specific information in the form of information elements.

In addition to the normal procedures we suggest the possibility for them to exchange authorisation statements in order to give additional credibility to their identities, although the parties could also take an opportunistic approach in the first stage and expect the latter parts of the process to provide more authenticity to their claims. Generally, the statements at this point ease the denial of service concerns to various parties of the whole interaction. So basically, the user could provide a statement, which identifies it as a subscriber of its home operator. Of course, if the identity of the home operator is unknown to the access operator, the statement has little value, unless additional certificate chains are presented that lead to a known trusted third party. The access network can also present a statement, which authorises it to provide access services at the current location. Again, the user might not find much value in this statement, if the identity is not known or any certificate chain is not presented. Note, however, that on certain contexts the user may have acquired this identity information through out of band means. For instance, the access might be available at some event, for which the ticket contains the identity of the operator (or rather, the hashed representation of it).

When the user and the access network have come to an agreement about the initial attachment, which at this point may only provide limited connectivity, the user can start the register process using a SIP REGISTER message. However, unlike in a typical case, this message also includes an authorisation statement, which contains the

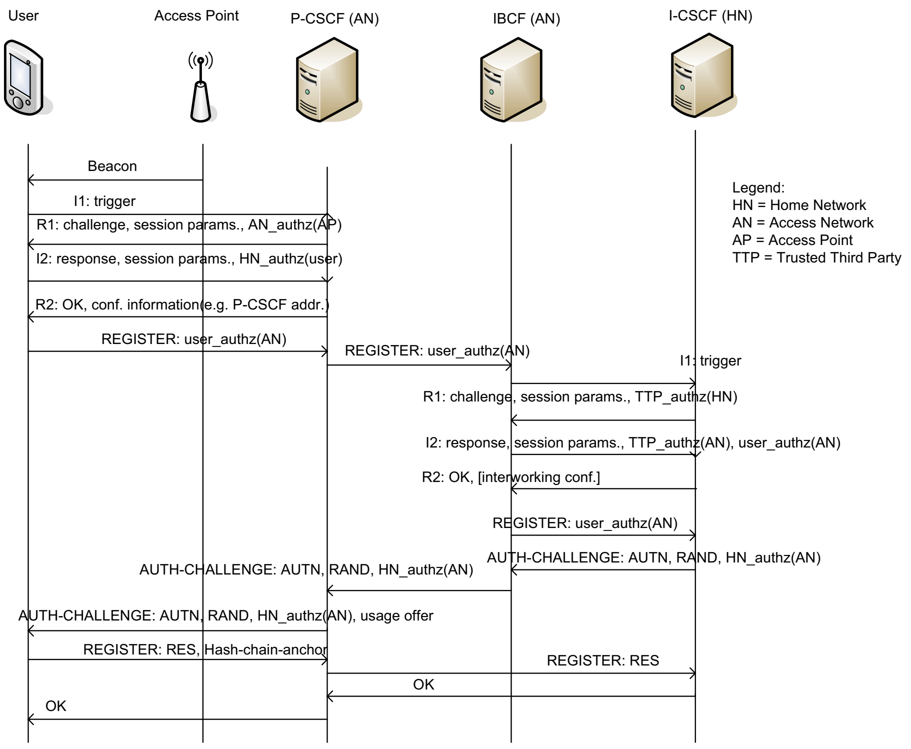

User
Access Point
P-CSCF (AN)
IBCF (AN)
I-CSCF (HN)
Beacon
I1: trigger
R1: challenge, session params., AN_authz(AP)
I2: response, session params., HN_authz(user)
R2: OK, conf. information(e.g. P-CSCF addr.)
REGISTER: user_authz(AN)
REGISTER: user_authz(AN)
I1: trigger
R1: challenge, session params., TTP_authz(HN)
I2: response, session params., TTP_authz(AN), user_authz(AN)
R2: OK, [interworking conf.]
REGISTER: user_authz(AN)
AUTH-CHALLENGE: AUTN, RAND, HN_authz(AN)
AUTH-CHALLENGE: AUTN, RAND, HN_authz(AN)
AUTH-CHALLENGE: AUTN, RAND, HN_authz(AN), usage offer
REGISTER: RES, Hash-chain-anchor
REGISTER: RES
OK
OK
Legend:
HN = Home Network
AN = Access Network
AP = Access Point
TTP = Trusted Third Party

Fig. 3. Message flows for securing the initial network attachment between the user and the access operator (AN) and establishing association between the access operator and the home operator (HN)

acknowledgement of the user that the access network is authorised to provide access to it. This is directed to the home network, so that it can tell that there really is its own subscriber connected to the particular access network.

The next step of the REGISTER message is dependant on the configuration of the network and typically P-CSCF would forward messages directly to I-CSCF of the home operator [9]. However, in this setting we assume that there is not yet any association, i.e. roaming agreement, between the two operators. Hence, the message is forwarded to the entities, which are better capable to negotiate on an operator level. P-CSCF of course could contain this functionality as well, but the security association is better established by border entities in order to be useful to other elements of the access network as well, i.e. some other P-CSCF of the access network could use the same security association to tunnel its traffic to the same home network.

The SIP REGISTER, as it is directed to the home network, contains enough information (along with Domain Name System, DNS) for the IBCF to know, with which entity to start negotiating. The procedure is the similar HIP like attachment, which negotiated the network level association. The authorisations are more important at this stage, because if there is going to be interaction between two operator level entities, they need more assurance about the trustworthiness of the other party, especially if there is going to be monetary consequences of the actions. Of course, there might be some previously gained knowledge about the identities, for which only proof of possession needs to be presented. Generally, though, the parties need certificates issued by entities, which they both are able to trust and ensure the liability, i.e. there is always some party, who will cover the costs. This could be some financial institution but also an organisation such GSM Alliance as there currently is no real global public key infrastructure (PKI). Note that the previously mentioned authorisation statement made by the user is used at this stage to give assurance to the home operator that there really is its real subscriber at the other end and there is motivation to engage in negotiation. Further note that in the figure we only show a simple handshake for security association establishment and it is not a full fledged negotiation as one might expect to happen between operators. This relates to the work done on composition and dynamic roaming agreements, but is not discussed here. Further information can be found, e.g., from [10]. The specifics of the negotiation procedure are also a direction for future work.

Once the network level security association has been established, the user registration at SIP level can proceed. Note that the user issued authorisation is still included so that the other IMS elements inside the operator core can act based on that information, such as the involved identities. While the whole core IMS topology is not shown in the figure, the process includes fetching the relevant subscriber information from the operator databases and issuing a challenge, which will authenticate the user as a real subscriber of the home operator. In a sense, the home operator already knows that there exists a user, who has made the statement about the access operator, but in order to ensure that the user is a live one and not some replay of some old message, the IMS AKA procedure is used. In addition to normal AKA information the message contains the assertion made by the home operator that it has negotiated with the current access operator. This way the user can know whether it is connected to the same operator, which negotiated with the home operator. This prevents man in the middle attack. P-CSCF also can include an offer statement. In other words, it states that it is offering services at a certain price and expects the user to acknowledge usage through hash chains, i.e. this will provide the relevant accounting information. For instance, it might require that in order to use the connection the user has to provide a new hash token every five minutes or after every 100 kilobytes. The offer could also relate to some local media service provided by the access operator, which would not involve the home operator in any way.

When the user receives the challenge it responds to it by calculating the expected response. Additionally, it includes a hash chain anchor to the message and binds it to its own identity using a signature. This way a user can use the subsequent hash chain values to “pay” for its use. The anchor is also bound to the offer and it is sent to the home operator as well in order to make sure that it is within the limits of the agreement the operators have made. The rest of the messages just acknowledge the process.

5 Hash Chain Based Service Usage

While the system described in the previous section can be used to set up the necessary identity relationships, the subsequent actions still need to be accounted in an assured way. This can be done with the help of hash chain tokens, which are securely bound to the identities employed in the setup phase.

The idea of using hash chains to provide accounting is not new; see, for instance, how it is used for micropayment in [11] and [12]. The security of the scheme is based on the properties of secure hash functions. In particularly, when you have a hashed value, you are not able to calculate the original value, the preimage, and neither are you able to calculate some other source value that results into the same hash value. So, when you first generate a random seed value, you are able to calculate a chain of *n* values by recursively employing the hash function, i.e. given the seed value *x* and hash function *H*, $x_n=H(H^{n-1}(x))$. The last value in the chain, i.e. the one that will be given to the other party first, is called the anchor value, and the preimages are used as tokens to represent the incremental payment of the service.

The idea is that the provider binds itself to the offer by signing it with its own identity. Next the user creates a hash chain anchor, which is bound to the identity of the user. This is also bound to the given offer, so it basically could be a hash of the offer statement. Note that at this point, the user and the access provider have already increased trust to their identities, because the home operator has acknowledged the user as its own subscriber and access network has been approved by the home network.

When the offer is bound to the provider, it cannot later repudiate its offer. Similarly, when the user is bound to the hash chain anchor, the user cannot deny having used the subsequent hash chain values to pay for the service. The incremental use of the values ensures that the loss will not be big, if the service is no longer provided: the user simply stops sending additional tokens. If the provider, on the other hand, does not receive any new hash tokens, it can just stop providing service. At the time of the clearing the provider can present the offer, the relevant hash chain anchor and the last received hash value. The clearing house (or the home operator) can check how many hash tokens were used and compensate the provider accordingly and later it will probably bill the user. The provider is not able to make unauthorised payment claims, unless it is able to break the hash function and create the next value of the chain.

The scheme can be realised with the help of Simple Public Key Infrastructure certificates [13], which can encode the offers and corresponding responses and provide the binding to the used identities. Other kind of certificates could be used as well, but SPKI is chosen because of its lightweight approach and illustrativeness in the scope of this work. An example of such certificate is given in Fig. 4. Note that the issuer and subject need to be identified, so that some other entity could not later make claims, if it for some reason would get hold of the transmitted hash chain values.

When releasing individual hash chain values, one needs a new SIP header for that. It also identifies the used hash chain, i.e. the same value that was given in the response to the offer. Additionally, the SIP message will contain the identity of the entity. This employs a mechanism specified in [14], which defines Identity and Identity-Info header fields. Basically, they just contain a signature and reference to the used identity. However, in the RFC the identity headers are inserted by the authentication server, whereas in our scheme the end entities are responsible for generating the signatures. An example of the SIP message headers are given in Fig. 5, which includes the corresponding identity and accounting fields. Note that not all the headers

```
(
 (cert
  (issuer (hash sha1 #c5dcb0ea158983ee8e5d922486259a72da5517c5#))
  (subject (hash sha1 #a80c2d93b476827fb75c2003e919f95914191bd8#))
  (offer (time 1 (s 60)))
  (validity (not-after 2008-07-30_12:00:00))
  )
 (signature (rsa-sha1 |Ccp+C2xOAg9fsBmNhQ4HHTftyFwTlw1k+KVMCjEqNXm
W5KSpUuEkTJMS5RKGuTkZSWHwrP0FA9MOcSS/wl+RtKMePYefXWpeNopsHmPzUUBC8
mRN0agwIeOoQ4AXqPgymlSMQoKoAGyL/AMXN7EE46nyFzmJpCB7rBXGe+DnutU=|))
)
```

Fig. 4. Example of SPKI certificate used to make an offer of service usage, i.e. give one hash token every 60 seconds

```
INVITE sip:userB_public@homeB.net SIP/2.0
From: <sip:userA_public@homeA.net>;tag=4fa3
To: <sip:userB_public1@homeB.net>
Contact: <sip:[5555::aaa:bbb:ccc:ddd];comp=sigcomp>;expires=600000
Date: Thu, 11 Sep 2008 13:01:03 GMT
Call-ID: apb03a0s09dkjdfglkj49111
CSeq: 101 INVITE
Identity:
"ZYNBbHC00VMZr2kZt6VmCvPonWJMGvQTBDqghoWeLxJfzB2a1pxAr3VgrB0SsSAa
ifsRdiOPoQZYOy2wrVghuhcsMbHWUSFxI6p6q5TOQXHMmz6uEo3svJsSH49thyGn
FVcnyaZ++yRlBYYQTLqWzJ+KVhPKbfU/pryhVn9Yc6U="
Identity-Info: <urn://subscribers.homeA.net/userB.cer>;alg=rsa-
sha1
X-Accounting-Info: id=4cac8d1a6ae4e715e80b3546bf44e776ed0be919;
value=049ce48c25c832dbc9efab84e389f675ebd0ae82
Content-Type: application/sdp
Content-Length: XXX

<sdp body follows..>
```

Fig. 5. Example of SIP message containing a hash token and identity info

are not shown, and because of the mutability of certain SIP headers, they are not all included in the signature calculation. From the signature point of view of interest are: From, To, Cseq, Date, and Call-ID. This allows identifying the correct senders and receivers as well as the used SIP session.

6 Evaluation

This work concentrates on integrating three different points on the selected environment. Firstly, it describes a solution for securely attaching the user to a visited network irrespective of the used access technology. Secondly, it shows how to take advantage of the initial access of the user to set up a security association between two operators, who do not have a previous relationship, i.e. they have not created a roaming agreement. Thirdly, as the parties are assumed to be previously unknown to each others, it outlines an accounting solution, which provides non-repudiation to the service usage so that the authenticated identities are bound to the accounting information. In other words, it is not possible at later stage to claim that the user has not used the service. Given the incremental approach for this accounting the user is also protected against the misuse of the provider.

The portrayed mechanisms are based on the assumption that every entity has a cryptographic identity, which prevents spoofing. This is not restricted just to network element, but it is expected that the network has an identity as well. For instance, the access network is considered to have an identity, which can issue statements of its own and which statements can be issued for. Thus, the identities are used in a cross-layer fashion, so that the actions on network and SIP level are intertwined and can be bound to the same assertions and acknowledgements.

The used network attachment procedure ensures the sameness property of the communicating partners. Thus, without any additional infrastructure, it is able to provide opportunistic authentication of the identities, so that it can be made sure that the communication partner has not changed, even though the real identity may not be known. An additional level of trust is gained either through the composition negotiation procedure or directly the authorisation statements, which rely on the existence of third parties, who has the trust of the both parties or, at least, has made arrangements to ensure the liability of the parties in case of actions that entail compensation related issues. This is especially needed in the operator level communication, otherwise the user and the home operator could collude against the access operator. Malicious access operator, on the other hand, has little to gain: if it does not provide service as promised, it does not receive any compensation, i.e. no hash chain values for accounting purposes. If it tries to act as a middle man and receive compensation instead of the legit access operator, the user will notice the difference between the identities, i.e. the one it has been talking to and the one that is authorised by the home operator after the negotiation process.

However, in the initial attachment between the access operator and the user, the authorisations may not be necessary, even though this may provide an avenue for denial of service attacks. This could happen, for instance, so that the adversary claims to be the local access operator, but denies the access to the user. This has the effect of defaming the innocent access operator in the face of the user. Malicious user can make the access user to initiate an unnecessary connection attempt toward the alleged home operator, but this does not go far, because the home operator quickly notices that the user in question is not its own subscriber. It is after all assumed that the home operator knows the identity of its customer, which is used to make the relevant assertions.

The use of hash chain based approach can be used to enhance the efficiency of the SIP signalling. In other words, it is possible to use a delegated approach, which does not require all the signalling to go through the home operator. This feature of IMS has the tendency to complicate the whole system, but it is understandable, because the home operator wants to be in control of its own customers and get accounting data of its own for every session. Through hash chains this accounting information can be provided in a non-repudiable fashion. Of course, it is good to keep in mind that this is also a political issue for the big operators, so just providing a technical solution might not always be acceptable for them. This could be, though, one of the things that are agreed during the dynamic roaming agreement negotiation. Note that while we are proposing a rather simple accounting system here, it could be further extended to take into account the micropayment work done for SIP, such as [15].

7 Future Work

In this paper we have outlined on a high level architecture for providing secure attachment and accounting properties for the IMS. Further work is needed to sort out the actual implementation implications and test the performance of the suggested techniques, especially regarding the overhead caused by the employment of cryptographic identifiers and certificates. There is also an issue with larger message sizes and their effect on packet fragmentation as discussed in [16]. It should be kept in mind, however, that this work is intended for future networks, even though all of the used mechanisms are available in various research prototypes, although not in the scale we are proposing here. More work is also needed on the concept of dynamic roaming agreements and the related negotiation mechanisms. One should keep in mind, though, that while it is possible to provide technical solutions, another thing is how acceptable they will be from the political and economical point of view.

8 Conclusion

We have depicted an architectural extension to the current IMS architecture in an environment where a user wishes to use access network, which does not have a roaming agreement with the home operator of the user. The proposal takes advantage of the cryptographic identities, which form the basis of the solution with the help of HIP like attachment procedure suited for heterogeneous environments. Those identities are assumed to represent the entities in such a way that they can be exploited in a cross-layered fashion for securing network and SIP level communication. This was further enhanced with the inclusion of authorisations, which allow the parties to make various statements about the characteristics of the other parties, such as proving that the communication has ensued with the intended entities. Thus, unlike today, every party is identified in a secure fashion.

In addition, to alleviate the concerns regarding the accounting information originating from a potentially untrustworthy source, the architecture includes the possibility of adding non-repudiable service usage to the subsequent signalling. This was done with the help of hash chains, which are securely bound to the identities of the communicating parties used in the setup phase. This way it is possible to provide assured accounting records, which can be used as a basis of billing. The incremental approach ensures that misbehaviour can be detected early on and the communication can be ended without fear of incurring extra charges or unnecessary burden on the service.

It is worth noting that the things like HIP are still years away from wide scale deployment, but this work is intended to show the various possibilities for securing the future networks.

References

1. Johnsson, M. (ed.): Draft System Description, Ambient Networks project deliverable D7-A (January 2007)
2. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T. (eds.): Host Identity Protocol. IETF Internet-Draft draft-ietf-hip-base-10, work in progress (October 2007)
3. Jokela, P., Moskowitz, R., Nikander, P.: Using ESP transport format with HIP. IETF Internet-Draft draft-ietf-hip-esp-06, work in progress (June 2007)
4. 3GPP. IP Multimedia Subsystem (IMS). 3rd Generation Partnership Project Technical Specification. TS23.228 V8.1.0 (June 2007)
5. Rosenberg, J., et al.: SIP: Session Initiation Protocol. IETF RFC 3261 (June 2002)
6. 3GPP. Security architecture. 3rd Generation Partnership Project Technical Specification, TS 33.102 V7.1.0 (December 2006)
7. Calhoun, P., et al.: Light Weight Access Point Protocol. IETF Internet Draft draft-ohara-capwap-lwapp-04, work in progress (March 2007)
8. Heikkinen, S., Priestley, M., Arkko, J., Eronen, P., Tschofenig, H.: Securing Network Attachment and Compensation. In: Proceedings of Wireless World Research Forum Meeting #15 (November 2005)
9. 3GPP. IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP). 3rd Generation Partnership Project Technical Specifica-tion. TS 24.229. V7.8.0 (June 2006)
10. 3GPP. Network Composition Feasibility Study. 3rd Generation Partnership Project Technical Report. TR22.980 V8.1.0 (June 2007)
11. Tewari, H., O'Mahon, D.: Multiparty micropayments for Ad Hoc Networks. In: Proceedings of the IEEE Wireless Communications and Networking Conference (March 2003)
12. Zhou, J., Lam, K.: Undeniable Billing in Mobile Communication. In: Proceedings of 4th ACM/IEEE International Conference on Mobile Computing and Networking (October 1998)
13. Ellison, C. (ed.): Simple Public Key Certificate. IETF Internet-Draft draft-ietf-spki-cert-structure-06.txt, expired (July 1999)
14. Peterson, J., Jennigs, C.: Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP). IETF RFC 4474 (August 2006)
15. Ruiz-Martínez, A., Sánchez-Laguna, J.A., Gomez-Skarmeta, A.F.: SIP extensions to support (micro)payments. In: Proceedings of 21st International Conference on Advanced Networking and Applications (May 2007)
16. Heikkinen, S.: Authorising HIP enabled communication. In: Proceedings of 10th International Symposium on Performance Evaluation of Computer and Telecommunication Systems (July 2007)

An Experimental Evaluation of a HIP Based Network Mobility Scheme

Jukka Ylitalo[1], Jan Melén[1], Patrik Salmela[1], and Henrik Petander[2]

[1] Ericsson Research NomadicLab, FI-02420 Jorvas, Finland
jukka.ylitalo@ericsson.com
[2] NICTA

Abstract. In this paper, the authors present and evaluate a network mobility scheme based on Host Identity Protocol (HIP). The cryptographic host identifiers are combined with an authorization mechanism and used for delegating the mobility management signalling rights between nodes in the architecture. While the delegation of the signalling rights scheme itself is a known concept, the trust model presented in this paper differs from the MIPv6 NEMO solution. In the presented approach, the mobile routers are authorized to send location updates directly to peer hosts on behalf of the mobile hosts without opening the solution for re-direction attacks. This is the first time the characteristics of the new scheme is measured in the HIP moving network context using a real implementation. The trust model makes it possible to support route optimization and minimize over-the-air signalling and renumbering events in the moving network. The measurements also reveal new kinds of anomalies in the protocol implementation and design when data integrity and confidentiality protection are integrated into signalling aggregation. The authors propose solutions for these anomalies.

1 Introduction

A cluster of hosts moving in the same geographic direction, like passengers in vehicles, can benefit of a network mobility solution. In a basic scheme, hosts are attached via a mobile router (MR) to the Internet. The mobile router dynamically changes its topological attachment to the Internet which results also in traffic flow re-directions. Depending on the applied trust model the mobility management signalling can be transparent or visible to the communicating end hosts. The existing MIPv6 NEMO[2] solution is based on a trust relationship and location update signalling between mobile routers and their home agents. Some MIPv6 NEMO related optimizations also apply the trust relationship between end hosts and their home agents. In this paper, the authors present a network mobility scheme utilizing trust relationship between mobile routers and peer hosts. This is the first time when the efficiency and performance of such a system is measured with a real implementation.

To mitigate the trust issues, this paper presents a network mobility solution based on the so called identifier-locator split approach. In a basic case, mobile routers are authorized by hosts in the moving network to send location updates to the peers. The required authorization is implemented using public key based host identifiers. The independent communication between mobile routers and peers results in over-the-air signalling and route optimizations.

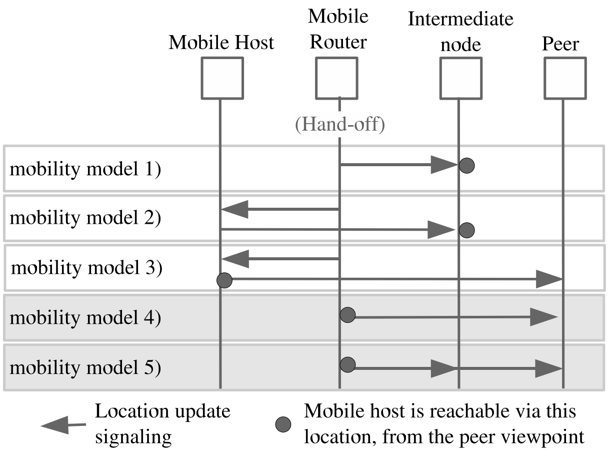


Fig. 1. Network mobility management models

The rest of this paper is organized as follows. The problem description is presented from the related work perspective in section 2. Section 3 describes the proposed network mobility scheme. The measurement results and related analysis are presented in section 4. Finally, section 5 concludes the paper.

2 Protocols for Network Mobility Management

Network mobility protocols enable sustaining of on-going connections between the mobile hosts inside a moving network and their communication peers in the fixed network. This goal can be achieved by using different signalling and routing models, as illustrated in Figure 1. The hand-off of the mobile router can be visible either to the mobile host, to the peer hosts, or to all of them. Depending on the approach, the peer host sends packets to an intermediate node (model 1,2), to the mobile router (models 4-5), or directly to the mobile host (model 3). Each model has different trade-offs between signalling latency, security vulnerabilities, the amount of control signalling and communications overhead.

The NEMO working group at the IETF has standardized a network mobility solution, NEMO [2], based on Mobile IPv6 which is based on model 1. In the MIPv6 NEMO approach, the mobility of the mobile router is hidden from both the mobile hosts and their peers. This results in so-called dog leg routing where packets are routed via an intermediate hop which causes additional end-to-end communication latency and increased communications overhead from tunneling of the packets. Further, the home agent may cause a bottleneck for the connections. However, NEMO allows for aggregation of mobility signalling. The packet forwarding path via the intermediate node protects the peer host from traffic re-direction attacks (see [1]), because it does not need to verify the new location of the mobile host (model 1). On the other hand, the resulted unoptimized routing path increases the end-to-end network latency.

Route optimization techniques based on model 1 have been presented by Ohnishi [11], Wakikawa et al [16] and Kang et al [5]. These techniques reduce the end-to-end communication latency by finding intermediate nodes closer to a direct route between the mobile router and the peer nodes. However, the communications overhead from tunneling remains the same as in NEMO.

Thubert et al [15], Jeong et al [4], and Petander et al [14] have proposed mechanisms based on model 3 which reduce the end-to-end communications latency and the communications overhead from tunneling. However, these mechanisms share the security weaknesses of Mobile IPv6 route optimization, making them vulnerable to on-path attackers. Further, since a handoff creates signalling for each mobile host, frequent handoffs combined with a large number of mobile hosts may lead to a signalling explosion. This may affect the on-going communication sessions negatively by temporarily reducing the bandwidth available to application data traffic. The model 4 is shortly analyzed in [7] from MIPv6 NEMO viewpoint.

The Host Identity Protocol (HIP) [9] enables secure communications between the mobile hosts and the peer hosts. HIP can be extended, as proposed by Ylitalo in [18], to enable network mobility based on models 4 and 5 through delegation of mobility signalling from the mobile hosts to a mobile router and finally to a signalling proxy that is located in the fixed network. The mobile router can then use the delegated authority to securely update the peer hosts of the location of the mobile hosts. The HIP network mobility protocol routes packets directly between the mobile router and the peers. The signalling required for the direct routing may lead to a signalling explosion in the same way as for the Mobile IPv6 based solutions. However, when public key cryptography and certificates are used for protecting the location update messages, like in [18][10][13], the effects of the signalling explosion are more severe than when low security route optimization mechanisms are used. Therefore the authors present a HIP based moving network scheme in [6] that uses symmetric cryptography in Kerberos-like fashion for delegating location update signalling rights between mobile hosts and mobile routers. In addition, Paakkonen et al make performance analysis of our HIP mobile router implementation in [13].

The work by Nováczki et al [10], represents a piece of independent work, published after the submission of this paper. It partially builds upon our previous work [8][18][12], reducing dependency on SPINAT[20] and going beyond our initial work, for example, through introducing the details for nested mobile routers. Most importantly, they have built an independent simulation model with measurements, which strongly support the validity of our approach.

3 Proposed Network Mobility Scheme

The design goals of the presented network mobility scheme have been minimizing the over-the-air mobility management signalling together with the re-numbering events without sacrificing the optimal routing paths. To achieve these goals, without opening the architecture to re-direction and Denial-of-Service (DoS) attacks (see [1]), HIP [9] was selected as a candidate protocol for implementing the delegation and aggregation of mobility management signalling rights between nodes. The proposed scheme is based on the mobility models 4 and 5 in Figure 1. The network mobility scheme discussed and evaluated in this paper is based on the initial ideas presented by Ylitalo [18] and Nikander et al [8]. In addition, according to Walfish et al [17] the delegation is a new primitive in the Internet architecture.

3.1 Host Identity Protocol (HIP)

HIP [9] supports secure host mobility and multi-homing; even between different address families. Each node generates an asymmetric key pair that works as a global Host Identifier (HI). The IP addresses define the topological location of the node in the network. The dynamic binding between HIs and locators results in the so called identifier-locator split approach.

The base HIP protocol consists of two kind of exchanges, namely base-exchange and update-exchange. The two round-trip base-exchange uses the HIs for mutual authentication. During the exchange the end-points establish security associations (SAs) between each other. The established SAs are used to protect the integrity and confidentiality of the Encapsulated Security Payload (ESP) packets. The keying material is also used to protect the three-way update exchange. The update exchange is used for mobility management signalling and re-keying. In addition, HIP supports a registration extension that is used by mobile nodes to request services, like a rendezvous service, from intermediate nodes.

The network mobility scheme, presented in this paper, combines the public key based host identifiers and authorization certificates (like SPKI[3]) for expressing trust relationships between nodes in the architecture. The self-signed authorization certificates provide a mechanism for delegating signalling rights between nodes. The authors believe that the presented solution can also be implemented with Cryptographically Generated Addresses (CGAs) in the MIPv6 NEMO context. The authorization can also be based on symmetric cryptography as presented in [6]. The main reason to evaluate the new scheme in the HIP context is its integrated data integrity and confidentiality protection. The results imply that this property also causes the biggest scalability problems (section 4).

3.2 Basic Network Mobility Concept

In our solution, the hosts in the moving network must support HIP protocol, while MIPv6 NEMO is backward compatible with legacy nodes. On the other hand, Figure 2 illustrates how HIP integrates end host mobility into network mobility in a seamless and secure way. Each communication session triggers a HIP base exchange with a new peer node (1 in Figure 2) and establishes a mobility states at end hosts for the later mobility. Once a mobile host joins the moving network, it initiates a MR discovery exchange. In this paper, the service discovery protocol is integrated with the end-to-end update exchange according to [12]. In practice, the end-to-end update exchange (2 in Figure 2) triggers an authorization and service registration exchange between the mobile host and the mobile router. The mobile host requests the signalling proxy and rendezvous services, and authorizes the mobile router to signal on behalf of it using authorization certificates (3 in Figure 2).

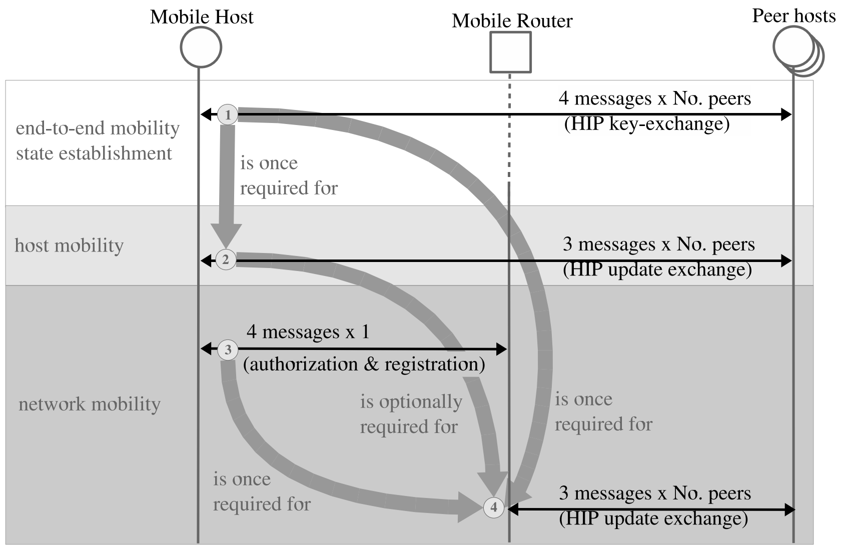

Mobile Host
Mobile Router
Peer hosts
end-to-end mobility state establishment
host mobility
network mobility
1
4 messages x No. peers (HIP key-exchange)
is once required for
2
3 messages x No. peers (HIP update exchange)
3
4 messages x 1 (authorization & registration)
is optionally required for
is once required for
is once required for
4
3 messages x No. peers (HIP update exchange)

Fig. 2. Basic mobility management signalling in the presented moving network scheme

Once the mobile router makes a hand-off, it sends location updates to the peer nodes on behalf of its registered clients (4 in Figure 2). In the basic case, the mobile router includes an authorization certificate to each of the location update messages. Each peer host verify that the certificate is signed by the host identity of an authentic mobile host and sends a challenge message back to the location claimed in the update message, i.e., so called reachability test. The challenge messages are destined to the the mobile router that is authorized to reply to the challenges on behalf of the mobile nodes. The basic HIP protocol provides a mechanism for hosts to figure out network failures, e.g., mobile router failures.

3.3 Signalling Optimization in Moving Networks

Each initial end-to-end update exchange transparently establishes a state at the mobile router.[1] The state at the mobile router is used for implementing a new kind of NAT functionality for IPsec ESP protected payload packets [20]. The NAT functionality at the mobile router provides a static and private locator space inside the moving network that solves the address allocation and re-numbering problem during hand-offs.

It is possible to aggregate the location update signalling to the mobile router in a secure way due to the public key based host authentication, certificate based authorization and the new NAT[20] functionality (1-3 in Figure 2). As a result, the mobile router is able to hide the locator changes from its clients. However, the mobile router should inform the clients about the hand-off event to allow them to adapt to the situation. The approach minimizes the re-numbering events and over-the-air mobility signalling in the

[1] The host may also run the base exchange through the mobile router for the same purpose.

moving network during hand-offs and provides a way to support route optimization that also were the main design goals. On the other hand, compared to MIPv6 NEMO solution the present solution increases CPU and bandwidth consumption at the mobile router during hand-offs affecting the battery lifetime.

From the scalability viewpoint, one drawback is that the mobile router must sustain a state for each HIP session flowing through it, unlike the MIPv6 NEMO solution that keeps a state per client host at the mobile router's home agent. Thus, the present route optimization scheme results in sacrifices in terms of hard states. The hard states at the mobile routers imply bigger hardware requirements than with the MIPv6 NEMO solution. From the trust model viewpoint, the peer host establishes a trust relationship with the mobile router that is used for location update signalling between those two entities. The situation is different in the MIPv6 NEMO case where the peer host does not need to trust the mobile router, because the location update signalling takes place between the mobile router and its trusted home agent.

From the nested moving network viewpoint, it is necessary to delegate the location update signalling rights between nested mobile routers to sustain communications sessions for the end hosts. Otherwise, the mobile router that is attached to the Internet is not able to send location updates on behalf of clients located in the nested moving networks. The presented solution does not require internal tunneling headers like MIPv6 NEMO in the nested moving cases due to the NAT functionality [20] at mobile routers. This saves bandwidth consumption and optimizes the payload packet sizes. On the other hand, the presented scheme must tackle the possible locator prefix collisions between the adjacent private networks.

3.4 Signalling Optimization between Mobile Routers and the Internet

Once the mobile host joins the moving network it authorizes the mobile router to send location updates on behalf of it. The mobile router may delegate the signalling rights to a signalling proxy that is located in the fixed network (1 in Figure 3). The mobile host/router also runs an update exchange with the peer nodes in parallel with the authorization exchange (2 in Figure 3). The present approach utilizes on-the-path signalling proxies. Distributing the signalling between multiple signalling proxies and routing the end-to-end update signalling through the alternative signalling proxies, e.g., using delegation between proxies or site multi-homing techniques at mobile routers is for further study.

The authorization between nodes can be expressed with a certificate chain that, in the present example, consists of two certificates. The mobile host authorizes the mobile router with one certificate and with the second one the mobile router delegates the location update signalling rights to the signalling proxy. When the mobile router makes a hand-off, it sends a single location update message per signalling proxy (3 in Figure 3). The signalling proxies runs a reachability test with the mobile router (4 in Figure 3). This triggers a burst of location update signalling between the signalling proxies and the peer hosts (5 in Figure 3). Both the mobile routers and signalling proxies should define a maximum number of updates per host identity to protect them from flooding and distributed Denial-of-Service (DDoS) attacks during hand-offs. The signalling proxies can use the available high bandwidth in the fixed network. The reachability test is run between the peer nodes and the signalling proxies (6 in Figure 3).

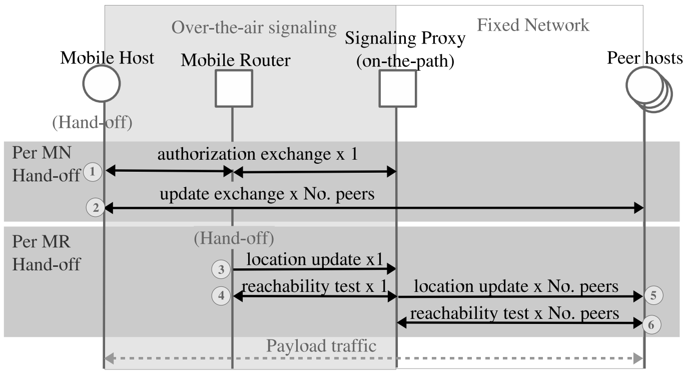


Fig. 3. Signalling proxy and the mobile router in the same trust domain

3.5 Signalling Optimization between Peers and the Internet

The peer hosts may also authorize a trusted signalling proxy to run reachability tests on behalf of them (1 in Figure 4). The approach optimizes the signalling between the peer nodes and the Internet. It is good to notice that the peers may also utilize wireless access technologies. From the mobile router viewpoint, the number of reachability tests can be reduced when multiple peer hosts authorize the same signalling proxy.

The peer hosts include the HI of the authorized signalling proxy to the initial end-to-end exchange (2 in Figure 4). In this way, the mobile host learns which of the peer hosts have authorized the same signalling proxy. Later on, when the mobile host joins the moving network, it authorizes the mobile router that further authorizes a signalling proxy in the fixed network (3 in Figure 4). The mobile host also sends in parallel location updates to the peer hosts' signalling proxies (4 in Figure 4). Each location update message contains a list of the HIs and ESP SPI values of the peer hosts that are located behind the same proxy. In this way, the mobile router is able to dynamically learn the location of the peer nodes. The on-the-path signalling proxies run reachability tests with the mobile host (5 in Figure 4), and inform the peer hosts about the mobile host's new location (6 in Figure 4).

Once the mobile router makes a hand-off it sends a location update to its proxy which then triggers a location update per proxy at the peer hosts side (7 and 9 in Figure 4). It is good to notice that the reachability tests (8 and 10 in Figure 4) are synchronized in a way that the mobile host's signalling proxy does not reply to the challenge message, i.e., sent by the peer host's proxy, before it has validated the mobile router's location. Typically, the latency between the mobile router and the mobile host's signalling proxy is shorter than the latency between the mobile host's and peer host's signalling proxies. Furthermore, the peers are informed about the mobile hosts current location (11 in Figure 4). The first payload packet destined to the mobile host works as an ACK for the received location update message.

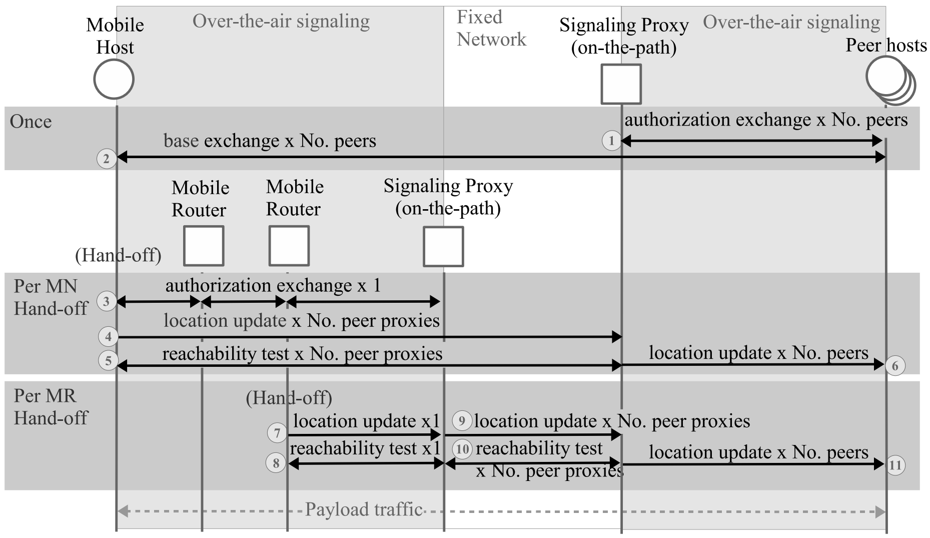

Mobile Host
Over-the-air signaling
Fixed Network
Signaling Proxy (on-the-path)
Over-the-air signaling
Peer hosts
Once
authorization exchange x No. peers
base exchange x No. peers
Mobile Router
Mobile Router
Signaling Proxy (on-the-path)
(Hand-off)
Per MN Hand-off
authorization exchange x 1
location update x No. peer proxies
reachability test x No. peer proxies
location update x No. peers
Per MR Hand-off
(Hand-off)
location update x1
location update x No. peer proxies
reachability test x1
reachability test x No. peer proxies
location update x No. peers
Payload traffic

Fig. 4. Combining the signalling optimizations

4 Measurements and Analysis

The presented over-the-air optimizations result in a burst of update signalling during moving network hand-offs. Depending on the optimization the burst takes place in topologically different locations. Therefore, the authors measured how the different number of parallel update exchanges affect the hand-off time in connected and connection-less communication session cases. Logically, the test simulates either the signalling between mobile routers and peer hosts (4 in Figure 2) or between multiple signalling proxies (9 and 10 in Figure 4).

Figure 5 illustrates the test environment which is running hip4inter.net HIP implementation. Each Vmware guest FreeBSD6 Operating System (OS) was running a HIP daemon supporting multiple HIs. The 100Mb Ethernet link between the analyzer and the router was divided into two Virtual LANs (VLANs). The router advertised different IPv6 prefixes through the VLANs. The hand-off was implemented by dynamically changing the bridging between VLANs and the interface at analyzer side of the mobile routers. During the hand-offs the CPU load was momentarily close to 100% and the available processing power was equally divided between quest OSs. The authors focused on the networking layer signalling. Therefore, the link layer related hand-off optimizations and TCP adaptation techniques are not included in the tests. Studying the behavior of alternative radio technologies is for further research.

The authors measured the mobile router's hand-off time, bandwidth consumption during hand-off and the amount of re-transmission for 64, 256, and 512 parallel communication sessions.[2] It is good to notice that the nodes are in the present approach identified with HIs, not with IP addresses.

[2] 4 HIs at 4 mobile hosts and 8 HIs at the 4 peer hosts resulted in $2*4*2*4 = 64$, $4*4*4*4 = 256$, and $4*4*8*4 = 512$ HIP sessions between end hosts.

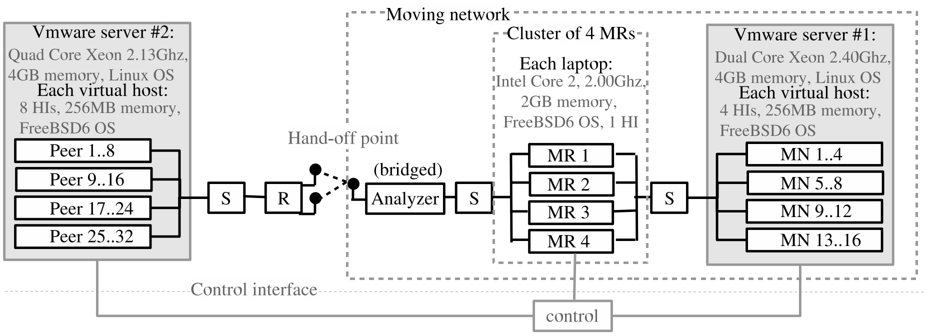


Fig. 5. Moving network test environment where switch (S), router (R), mobile router (MR) and mobile node (MN)

Figure 6 illustrates mobile router hand-offs for the different amount of communication sessions. From the congestion control viewpoint, the totally different TCP and ICMP traffic models were selected to figure out the strange behavior of the implementation. The *Netperf* (www.netperf.org) application was used for generating 64 TCP connections between end hosts. The scenarios of 256 and 512 TCP connections resulted in connections lost due to long hand-off times. Therefore, the connectionless *ping6* application was used to obtain measurement results for 256 and 512 communication sessions. The zero time is bound to the the first outgoing update exchange message in each chart in Figures 6 and 7 .

The order of the parallel update exchanges in Figure 6 is defined by the first incoming payload packet per session. The bottom line of each chart defines the initial delay before an update exchange started. The second line from the bottom illustrates the time when each 3-way HIP update exchange was completed from the mobile router viewpoint. The third line in the charts presents the time when the first payload packet was sent per session, while the uppermost line presents the time when the first payload packet was received. It also defines the hand-off time per session. The incoming payload packet of the last update exchange defines the total hand-off time for all the sessions. The corresponding values are presented in Table 1 and bandwidth consumption illustrated in Figure 7.

Table 1. Specified delays and overhead for the different phases of a hand-off procedure

Number of sessions: Payload type:	64 TCP	64 ICMP	256 ICMP	512 ICMP
Initial triggering delay for the update exchange (per session)				
Average:	0.025s	0.025s	0.102s	0.206s
95% limit:	0.048s	0.048s	0.195s	0.391s
3-way update exchange hand-shake time (per session)				
Average:	0.169s	0.171s	0.595s	1.176s
95% limit:	0.328s	0.328s	1.079s	2.139s
Delay between the last update exchange message and the 1st received payload packet (per session)				
Average:	1.415s	0.626s	2.831s	6.160s
95% limit:	2.842s	0.928	3.529s	7.996s
Hand-off time for 100%, 95%, and 50% of sessions.				
Avg. for 100% of sessions:	7.012s	1.370s	5.192s	10.792s
Avg. for 95% of sessions:	3.070s	1.250s	4.734s	10.434s
Avg. for 50% of sessions:	1.608s	0.822s	3.529s	7.542s
Average overhead of re-transmissions (per update exchange message type)				
Update message:	52%	54%	149%	199%
Challenge message:	52%	54%	151%	201%
Response message:	52%	54%	151%	201%

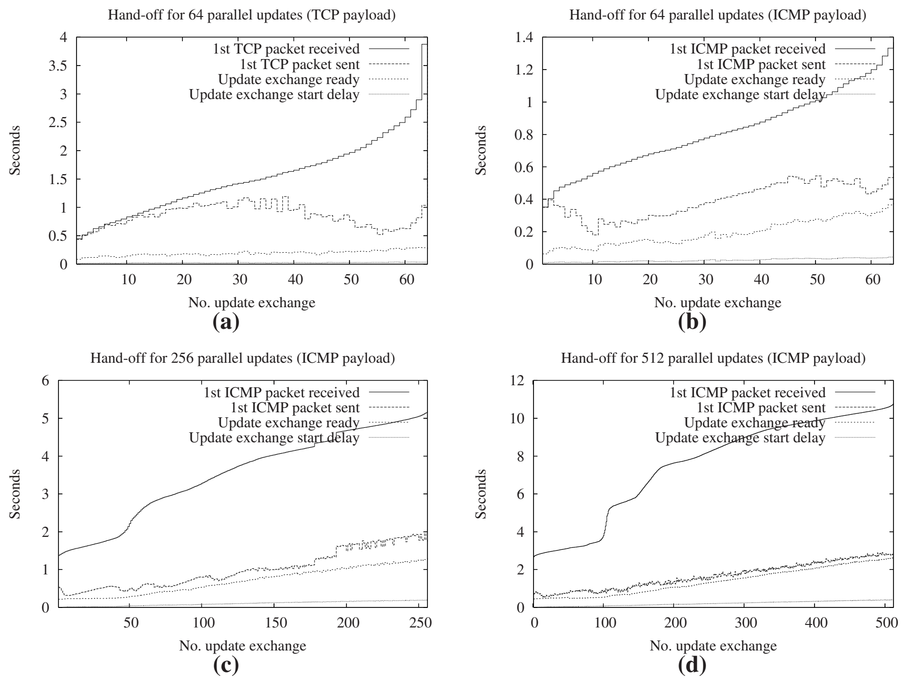


Fig. 6. Average hand-off times for 64, 256, and 512 end-to-end communication sessions at the mobile router. The hand-off in (a) is measured with TCP payload traffic. (b)(c)(d) are measured using ICMP payload traffic where the 'echo request' period is 400ms per communication session.

4.1 Analysis of Results

Overall, the results indicate longer hand-off times than the authors expected when there are high amount communication sessions. This section analyzes reasons for that and proposes improvements. The results indicate close to linear dependency between the number of sessions, total hand-off time (Figure 6) and the bandwidth consumption (Figure 7). Thus, dividing the signalling into small enough bundles between multiple signalling proxies results in a faster completion of the parallel update exchanges.

A substantial observation is that the maximum bandwidth consumption peak of the mobility signalling is almost the same in the different scenarios (Figure 7). In other words, the peak does not increase as a function of sessions. The main reason for this and for the initial update exchange triggering delay is the event based HIP daemon implementation. In other words, the packets are processed serially per HIP daemon and the daemons are not able to process the signed and HMAC protected location update messages faster. It is also good to notice that the mobile routers and guest OSs were running HIP daemons in parallel. Now, increasing the level of parallel update message processing at daemons and distribution of HIs between multiple hosts would decrease the initial delay and the processing time of the update messages in the presented test environment.

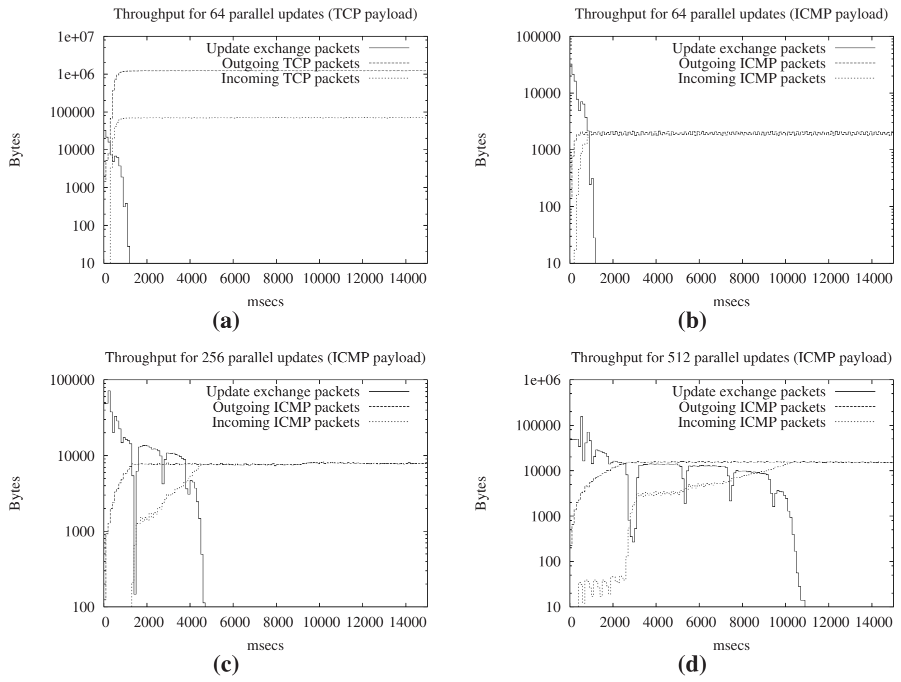


Fig. 7. Average bandwidth consumption during the mobile router hand-off for 64, 256, and 512 end-to-end communication sessions. Bytes per 100msec.

Another interesting observation in the measurements was the remarkable difference between the update exchange time and the incoming data packet waiting time causing the biggest delay during the hand-off procedure. The main reason for the behavior can be found by analyzing the authors' earlier work in [19] and the FreeBSD6 kernel implementation. The results in [19] show that a hand-off for a single HIP session results in an average of 143ms data packet waiting time (without additional network latency). This delimits the cause of the extra delay to the last procedure that takes place after the protocol has requested the kernel to update the IPsec policies and SAs.

Based on the authors' best understanding of the FreeBSD kernel behavior, the substantial incoming data traffic at the IP layer slows down the I/O communication between the user-space HIP daemon and the kernel IPsec module. The incoming IP packets cause interrupts at the kernel that go ahead of the IPsec socket API calls in an unfair manner.

Basically, the peer hosts are not able to send payload data back to the mobile hosts before the new IPsec policies and SAs are established. The results also strengthen the authors' viewpoint of the anomaly related to the kernel behavior. When the hand-off time and the bandwidth consumption are analyzed in parallel it is visible that the waiting delay for incoming packets increases in the function of outgoing data packet bandwidth consumption (Figures 6 and 7). In the TCP case (Figure 6 (a)), it is visible that incoming data packet delay correlates with the 3-way update exchange time, not with the outgoing data packet time. The total hand-off time is significantly increased by the last 5% of the TCP connections when the link is almost flooded.

Another observation is related to the unnecessary re-transmissions after all sessions have been re-routed to the new location. The reason for most of these re-transmissions is the processing delay of the parallel update messages. The drawback is that the overhead in the re-transmission increases the bandwidth consumption during hand-offs. The result indicates that it is important to implement a scheduling algorithm for re-transmission timers to optimize the packet processing delay and the overhead of re-transmissions. Basically, the mobile routers and the signalling proxies should adjust their re-transmission timers based on the number of parallel updates running. They should also limit the number of parallel update exchanges to the level they can process within a single timeout.

5 Conclusions

In the presented solution, the gained benefit from over-the-air and route optimizations depends on the level of parallelism in the mobility management signalling. To minimize the dependencies between the parallel update exchanges and increase parallelism during hand-off, a preferable instantiation of the presented scheme distributes the signalling between multiple signalling proxies. In addition, the results indicate that the current implementation does not scale well in terms of hundreds of parallel communications sessions. However, the implementation is based on the first presented signalling optimization in the moving network. To improve the measured hand-off times and to minimize the total number of required location update exchanges, the implementation must be extend to support signalling aggregation to the fixed signalling proxies at both sides.

Acknowledgements

The authors would like to thank Pekka Nikander, Raimo Vuopionpera, Yuri Ismailov, Jan Holler, Goran Selander, Goran Schultz, Mikko Sarela, Jari Arkko, Christian Vogt, Kristian Slavov and Petri Jokela for feedback and fruitful discussions related to the topic.

References

1. Aura, T., Roe, M., Arkko, J.: Security of Internet Location Management. In: Proc. of the 18th Annual Computer Security Applications Conference, Las Vegas, USA (December 2002)
2. Devarapalli, V., Wakikawa, R., Petrescu, A., Thubert, P.: RFC 3963: Network Mobility (NEMO) Basic Support Protocol (January 2005)
3. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: RFC 2693: SPKI Certificate Theory (September 1998)
4. Jeong, J.P., Lee, K., Park, J., Kim, H.: ND-Proxy based Route and DNS Optimizations for Mobile Nodes in Mobile Network (February 2004)
5. Kang, H., Kim, K., Han, S., Lee, K.-J., Park, J.-S.: Route Optimization for Mobile Network by Using Bi-directional Between Home Agent and Top Level Mobile Router. In: Internet-Draft, work in progress (June 2003)

6. Melén, J., Ylitalo, J., Salmela, P.: Host Identity Protocol based Mobile Router (HIPMR). Internet-Draft, work in progress (March 2008)
7. Ng, C., Zhao, F., Watari, M., Thubert, P.: Network Mobility Route Optimization Solution Space Analysis. RFC 4889 (July 2007)
8. Nikander, P., Arkko, J.: Delegation of Signalling Rights. In: Proc. of the 10th International Workshop on Security Protocols, Cambridge, UK, April 2002, pp. 203–212 (2002)
9. Nikander, P., Ylitalo, J., Wall, J.: Integrating Security, Mobility, and Multi-homing in a HIP Way. In: Proc. of the NDSS 2003, San Diego, CA, USA (February 2003)
10. Nováczki, S., Bokor, L., Jeney, G., Imre, S.: Design and Evaluation of a Novel HIP-Based Network Mobility Protocol. JOURNAL OF NETWORKS 3(1) (January 2008)
11. Ohnishi, H., Sakitani, K., Takagi, Y.: HMIP based Route optimization method in a mobile network. Internet-Draft, work in progress (October 2003)
12. HIP Service Discovery. Internet-Draft, work in progress (June 2006)
13. Paakkonen, P., Salmela, P., Aguero, R., Choque, J.: Performance Analysis of HIP-based Mobility and Triggering. In: Proc. of the 9th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM 2008), Newport Beach, CA, USA (June 2008)
14. Petander, H., Perera, E., Lan, K., Seneviratne, A.: Measuring and Improving Performance of Network Mobility Management in IPv6 Networks. IEEE Journal on Selected Areas of Communications, Special Issue on Mobile Routers and Network Mobility (2006)
15. Thubert, P., Molteni, M.: IPv6 Reverse Routing Header and its application to Mobile Networks. Internet-Draft, work in progress (February 2007)
16. Wakikawa, R., Koshiba, S., Uehara, K., Murai, J.: ORC: Optimized Route Cache Management Protocol for Network Mobility. In: Proc. of the 10th International Conference on Telecommunications (ICT 2003), French Polynesia, February 2003, pp. 1194–1200 (2003)
17. Walfish, M., Stribling, J., Krohn, M., Balakrishnan, H., Morris, R., Shenker, S.: Middleboxes no longer considered harmful. In: Proc. of the USENIX OSDI, San Francisco, CA, USA, December 2004,
18. Ylitalo, J.: Re-thinking Security in Network Mobility. In: Proc. of the NDSS Wireless and Security Workshop, San Diego, CA, USA (February 2005)
19. Ylitalo, J., Melén, J., Nikander, P., Torvinen, V.: Re-thinking Security in IP based Micro-Mobility. In: Proc. of the 7th Information Security Conference (ICS 2004), Palo Alto, CA, USA, September 2004, pp. 318–329 (2004)
20. Ylitalo, J., Salmela, P., Tschofenig, H.: SPINAT: Integrating IPsec into Overlay Routing. In: Proc. of SecureComm 2005, Athens, Greece (September 2005)

Service Discovery Framework for MANETs Using Cross-Layered Design

Balaji Raghavan, Jarmo Harju, and Bilhanan Silverajan

Department of Communications Engineering
Tampere University of Technology
Tampere, Finland
balaji.raghavan@ieee.org, jarmo.harju@tut.fi,
bilhanan.silverajan@tut.fi

Abstract. Service discovery for computer networks has traditionally been done in the application layer. An explosion of growth in the adoption of wireless technology has led to the emergence of a new breed of networks, mobile ad hoc networks (MANETs). These networks are constrained for resources, error prone and highly volatile. Cross-layered design approaches address the performance problems faced by traditional approaches to service discovery in MANETs. In these approaches the optimization is achieved by coupling the service discovery functions with routing functions, which reduces the generality of the solutions. In this paper we present a cross-layered solution to this problem that aims to preserve the best of both the traditional application layer and cross-layered design approaches. We propose a framework for service discovery in MANETs that not only exhibits superior performance but is also feature rich, takes an integrated approach to service discovery and is based on a modular design.

Keywords: Ad hoc networks, service discovery, cross-layered design, framework, broadcast mechanisms, link layer.

1 Introduction

Wireless networking over the recent years has emerged as a rapidly growing and popular technology for enabling the connectivity of mobile computing devices. Wireless networks support the concept of pervasive computing which enables instant connectivity and provides processing capacity which is not restricted by the limitation of geographical spaces. MANETs that can be formed without the need for any permanent infrastructure installation are more commonplace than ever. New and emerging fields of research such as cross-layered protocol engineering have exploited the opportunistic nature of wireless communication and addressed the challenges faced by applications deployed on mobile wireless networks

Cross-layered protocol engineering is a novel approach to software design for wireless networks which relies on significant interactions among various layers of the protocol stack [1]. Applications of such approaches for performance optimization are needed to address the challenges faced in designing solutions for these networks. However care has to be taken not to belittle the importance of sound architectural design as outlined in [2]. This paper outlines the design of a framework for service discovery in MANETs that combines the best features of existing solutions used in wired and wireless networks.

2 Related Work

Networks are formed for the purpose of sharing resources and an important step in enabling this is the discovery of services and the entities offering these services. Current solutions for service discovery in wired networks are based on approaches, which have traditionally adhered to considering service discovery at the application layer level. Many frameworks and protocols have been specified and developed for the purpose of automating the service discovery process. While some are geared towards a specific programming language or a specific platform, others offer broadcast or multicast as well as advanced support for filtering, querying and browsing. Some examples are Jini [3], SLP [4], DNS-SD [5], UDDI [6] and SSDP [7].

Service discovery mechanisms for wired networks do not work well in MANET environments. Due to the complexities of MANETs many lightweight service discovery protocols have been specified especially for addressing the problems of the ad hoc environment. The common goals of these protocols have been to minimize the overhead caused by the protocol, to be adaptive to dynamic environments, to be responsive to changes in the environment and to be flexible in their operation. Recent developments in the wireless research community have shed light on the willingness to take into consideration cross-layer protocol engineering for optimizing the performance of protocols employed in MANETs. So currently there are both cross-layered solutions and traditional lightweight application layer service discovery mechanisms available for MANETs. Some examples are KONARK [8], LSD [9], AODV-SD [10], service discovery extensions to ODMRP [11] and GSD [12].

3 Motivation

Service discovery protocols, which operate in the application layer of the OSI reference model, are scalable, flexible and secure and support zero-configuration and sophisticated querying capabilities. However they suffer from significant overhead and consume a considerable amount of resources due to their operation in the application layer. Cross-layered solutions, which integrate service discovery with routing functions, do not have such drawbacks. However, they take a minimalist approach to service discovery and do not espouse the rich feature set of the application layer solutions. There was perceived a need for providing a service discovery solution for MANETs that is both architecturally sound and feature rich as traditional solutions as well as superior in performance like the cross-layered solutions.

4 Service Discovery Framework

A service discovery framework for MANETs is proposed in this paper that is able to support discovery of services over different wireless connection technologies such as

IEEE 802.11, Bluetooth/IEEE 802.15 that are commonly used in the formation of MANETs. It is flexible and provides support for both discovery and interoperability between servers and clients operating in different types of networks. It is designed to be easily extendable for the purposes of scalability and security. It also optimizes the utilization of the resources in the MANET. It incorporates the best features of already existing service discovery mechanisms. The framework defines the environment, roles and mechanisms for service discovery in MANETs.

4.1 Architecture and Design

The architecture of the service discovery framework is as shown in Fig. 1. It consists of different components working together to support service discovery in a cost effective and optimal manner in the MANET. Most of the component functions in the framework are related to the process of service discovery but certain components such as the gateway also serve the purpose of providing seamless service access. The specification of the framework is flexible enough to be able to function in the absence of many of these entities. However maintaining a proper ratio of different components that contribute to the framework's functions to those that simply utilize its services significantly enhances the performance of the framework.

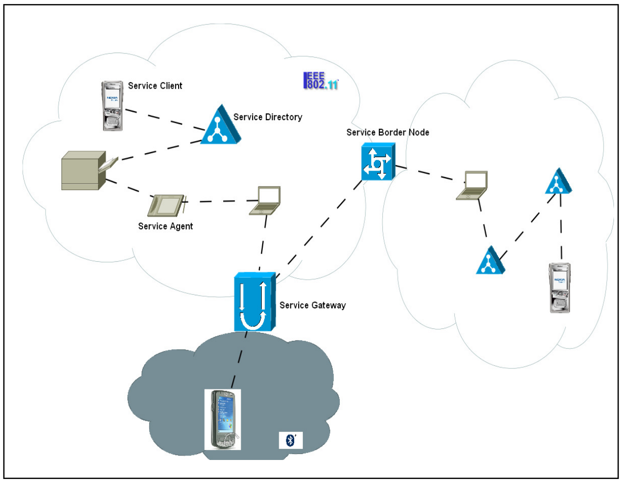


Fig. 1. Service discovery framework architecture

Different types of services can be offered in MANETs such as printing services, fax services, media services (music, video), voice over IP (VOIP), communication services such as conferencing and sensor statistics (temperature, pressure) collection. Service information is transformed using hashing to minimize its footprint in service discovery PDUs (Protocol Data Units). Each of the framework components has a service cache where it stores this configuration information along with other service discovery related data. Even though the framework requires some amount of caching, depending on the role assumed by the framework component and the resources available, this caching could be totally avoided.

The framework employs different protocols for the purpose of supporting service discovery. At the heart of the design of the framework is the splitting of the different aspects of service discovery between the application and link layer. The different functions involved in service discovery are supported by various protocols operating in the framework components. The application layer functionality is composed of many protocols used for configuration, and querying functions. The link layer protocol supports service advertisement, service requests (simple querying) and service teardown by utilizing the broadcast mechanisms of the link layer. The application layer protocol is used in the framework to support a rich variety of features that are designed to be flexible, scalable and extendible. The protocol sub-components, which operate in the link layer, are designed for the purposes of increasing the responsiveness and speed of the framework in resolving service requests.

4.2 Framework Components

The following is a description of the different components needed for the operation of the framework. The framework utilizes the different services provided by these components for performing the various functions of the framework. This modularization of the different functional aspects of service discovery into different components enhances the flexibility, extendibility and scalability of the framework. The different framework components are as follows:

4.2.1 Service Discovery Agents

All MANET nodes actively participating in the service discovery framework of the network are termed Service Discovery Agents (SDA). Each agent must have the facility to store and forward service requests and advertisements. They must implement some form of service caching however limited, for the purposes of service discovery. There must be at least one SDA within communication range of a framework component in the MANET for the proper operation of the framework.

4.2.2 Service Clients

Service Clients (SCs) utilize the services of the service discovery framework without actively participating in the functions of the framework. These nodes do not have any caching facilities and merely issue service requests. A large percentage of the framework population is comprised of the SDAs and clients clustered around them.

4.2.3 Service Directories

Service directories (SDs) are ideally MANET nodes which are not constrained by the limitations of limited memory, processing power and battery lifetime and can act as large caches of service information in the network. It is not necessary to have service directories for the operation of the framework but their presence definitely enhances the performance of service discovery within the framework. The SDs are also used for configuring the newly joined MANET nodes so that they can utilize the services of the framework. Another important function of the SDs is the registration of new services. SDAs that want to offer services which are not statically defined in the framework have to register these new services with an SD. The SDs may also work in conjunction with the Service Border Nodes (SBNs) to support exchange of service information with other MANETs.

4.2.4 Service Gateways

Service Gateways (SGs) support interoperability between servers and clients operating in MANETs using different wireless connection technologies and different protocols. Similar to the SDs these nodes are not ideally constrained by the resource limitations commonly associated with MANET nodes. These nodes act as service data routers and support caching and forwarding of service traffic between servers and clients located in different MANETs. It is not necessary for the presence of service discovery gateways unless the MANET wants to support interoperability services.

4.2.5 Service Border Nodes

SBNs also function like routers in a communication network but they only store and forward service discovery information between different MANETs. They can perform aggregation of service information and may also optionally provide routing information about the servers in different networks. The framework does not necessarily require the presence of service border nodes unless the MANET wants services from other MANETs to be accessible by its clients and agents. SBNs can also be used between MANETs employing entirely different networking architecture.

4.3 Framework Services

The different services provided by the framework are explained below. The different functions of the framework components in supporting the services are also elaborated upon.

4.3.1 Service Configuration

Service configuration is an essential function of the service discovery framework that is initiated when a MANET node joins the framework for the first time. The configuration function initializes the service cache of the nodes with essential information on different services available in the framework. Service configuration function is done by the means of an application layer protocol component in the framework. During the process of service configuration the configuration server (an entity resident in the SD) supplies information about the services which are available for discovery and access within the MANET. Service configuration for SCs and SDAs can also be done manually in the absence of the SD.

Service information is stored in the service cache of the framework components. Services in the framework are modeled using a serial number, protocol identifier, port identifier and other service attributes encoded in XML. A common service representation and identification scheme has to be used by all the participants of the framework. The service discovery framework supports both static and dynamic configuration in a MANET. The framework and its components reserve, allocate and associate service serial numbers for standard and commonly found services in MANETs. The framework supports dynamic configuration by reserving a range of service identifiers for this purpose and authorizing the SDs to perform this function.

In order to limit the amount of service information transmitted in the MANET during the process of service advertisement and discovery, services are identified by the means of a hashed service identifier. The hashed service identifier is generated from the service related information present in the cache such as serial number and attributes. This is done by the means of a cryptographic hashing function such as SHA-256 [13], which generates a service identifier hash of a definite length (16 bytes in the case of SHA-256) from service information.

4.3.2 Auto-configuration

Auto-configuration for the framework components is provided by a configuration protocol in the application layer. Auto-configuration of entities performing different roles in the network is done by subsets of the auto-configuration protocol. There are subsets of the protocol defined for SDAs, SCs, SGs and SBNs with different PDUs and mechanisms. Auto-configuration is used to set up the operation of different framework components when they join the framework for the first time. For example it is used in selecting a hashing function for service configuration, to indicate to the framework components that the transport layer used in the MANET is IPv4 or IPv6, specifying querying protocols to be used and for selecting other PDUs, protocols and mechanisms for providing the framework services. The auto-configuration service is provided by configuration server entities which are resident on the SDs. The SDs periodically advertise this service to other framework components in the network. In the absence of an SD there is no support for auto-configuration and the components have to be manually configured before they can join the framework.

4.3.3 Interoperability

Interoperability between servers and clients operating in MANETs using different connection technologies and protocols is provided by the means of SGs. Clients communicate with the SGs as they would communicate with a server and the SG re-routes the service traffic to the appropriate server. The server may operate on a MANET using a different connection technology than the one on which the client is operating. Two different modes of SG operation are supported by the framework, namely transparent and encapsulated operation. In the transparent mode of access the SG behaves as a proxy server, accepting service traffic on the same port using the same protocol as the real server would. In the encapsulated mode of operation the service traffic is encapsulated as payload in all interactions between the client, SG and the real server.

4.3.4 Querying Mechanisms

Sophisticated querying mechanisms are supported by the SDs and any other framework component which supports such a facility. Querying capabilities are also advertised as a service in the network and are provided by the SDs, SBNs and SGs. A common querying language is used in the framework components and is selected during the process of configuration. XQuery [14] is the default querying language used in the network. XQuery is the standard XML querying language developed by W3C which supports both simple and complex queries. XQuery was chosen as a default querying language in the framework because the service attributes are represented using XML.

4.4 Framework Features

The different features of the framework such as scalability, flexibility and resource optimized operation are explained in this section. Scalability in the framework can be provided by the presence of SBNs and by the use of a hierarchical service classification scheme for service identification. By using a service classification scheme that assigns service serial numbers to services in a hierarchical fashion the framework can achieve scalability by simple level-wise aggregation at the SBNs. The SBNs and service directories can also employ sophisticated caching, directory exchange and filtering as described in [15]. The SBNs and SDs within a MANET are constrained to utilize a common service identification/classification scheme for interoperability. The framework is flexible and adaptable to different environments by employing cross-layered design. It reduces the tight coupling with the routing layer when compared with existing cross-layered solutions. The components are loosely coupled with the link layer and do not interfere with the normal operation of other layer functions. This makes it easy to extend or to change the design of the framework.

The framework aims to minimize the utilization of resources in the MANET by reducing the amount of traffic due to service discovery. This it achieves by trying to minimize the control traffic overhead usually associated by performing service discovery in the application layer. Also by limiting the periodic broadcast of service information the framework can dynamically reduce the resource consumption in the MANET. Since service information is cached at framework components that possess the resources for such caching, service discovery can be done in a reasonably optimal and flexible manner even though the service advertisements are reduced.

5 Simulation and Analysis

The core components of the framework design were evaluated by implementing them in the NS-2 network simulator [16]. The 802.11 MAC layer implementation of the NS-2 network simulator was extended to support a subset of synchronization and scanning service. The beacon and probe request frames were extended to include the link layer service discovery advertisements and requests respectively. The service cache and an interface to the service cache were implemented. The advertisement algorithm was implemented so that there are periodic service advertisement broadcasts using the extended beacon frames. The beacon interval was set to 200ms which models typical settings in most 802.11 based MANETs. The service requests were broadcast using probe request frames and the replies were propagated using both the broadcast mechanism provided by the beacons as well as by an application layer PDU unicast to the originator of the request. The physical layer parameters for the 802.11 MANET were set to imitate 914 MHz Lucent Wavelan DSSS radio interface. The simulations were run for 600 seconds and results were averaged over 5 simulation rounds to obtain statistically reasonable estimates.

The MANETs simulated consisted of 50 or 100 nodes, each of them connected in the IBSS mode of an IEEE 802.11 radio interface so that they form a MANET. The area of simulation for the 50 nodes was a 1000mX1000m square area and for 100 nodes was 1500mX1500m.sq. The mobility of the nodes was modeled according to the random waypoint model. The number of services in the MANET was varied between 5 and 10 services and the nodes were limited to offering one service per node. The number of service requests in the MANET was varied between 50,100,150 and 200 for each simulation round. The service requests were uniformly distributed over the entire time of the simulation and among all the nodes of the MANET. All the MANET nodes in the simulation were SDAs which cached all the service advertisements they received. The SCs only contribute a negligible constant delay to the performance as they have to be in the vicinity of an SDA to issue a request and hence were not included in the simulations.

The operation of the framework under varying mobility conditions was also evaluated. The mobility can be estimated by observing the number of link changes and these were varied between ~1000 (low mobility), ~2100 (medium mobility) and ~6000 (high mobility) over the entire duration of simulation (600 seconds). All of the simulation rounds were done using the medium mobility pattern except for the ones in which the performance of the framework under varying mobility patterns was evaluated. The parameters that were considered for the evaluation of the framework were delay and traffic overhead. Delay refers to the delay in seconds experienced by nodes in discovering services using the framework over a fixed time period. Traffic overhead signifies the overhead caused due to the functioning of the framework in the MANET. The overhead can be estimated from the number of beacons transmitted. In the simulations 13.33% of the beacon traffic was contributed by the link layer service discovery extensions (i.e. around 7 bytes for each beacon frame).

The graph in Fig. 2 depicts the average delay experienced by the MANET nodes in discovery of services using the framework. The delay is plotted against the number of service requests for four different MANET configurations. The lowest delay experienced by the nodes was observed in MANET configuration in which there were 50 nodes and 5 services offered. The average delay for 50 requests is very low because-most of the requests are made after the service caches mature. For the 100 and 150 request cases the delay increases when compared to the delay for 50 requests, as more requests are made before the caches mature. As the number of service requests increases further i.e. to 200 the delay decreases sharply as more of the requests can be satisfied from the cache.

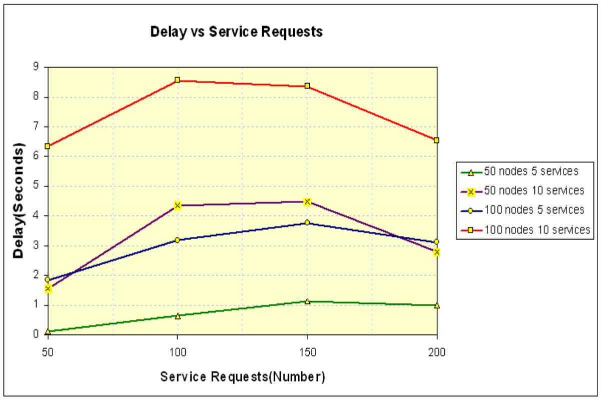


Fig. 2. Delay vs service requests

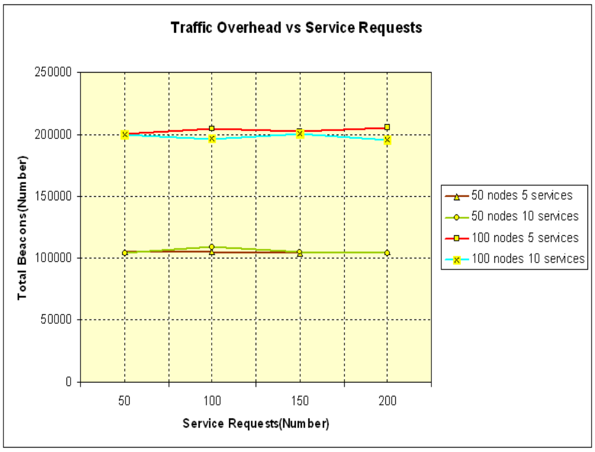


Fig. 3. Traffic overhead vs service requests

The graph in Fig. 3 depicts the total overhead caused due to the operation of the framework. This can be estimated from the number of beacons used for service discovery purposes. In the 50 nodes case the traffic overhead due to service discovery is around 685 kilobytes for the 600 seconds of simulation (100,000 x 7 bytes), which is *negligible* when compared to the data rates supported by 802.11. Because the number of beacons successfully transmitted during the duration of the simulation is fairly constant the traffic overhead incurred due to service discovery is also constant as shown in Figure 3. The number of beacon transmissions depends upon the number of MANET nodes and the mobility pattern used.

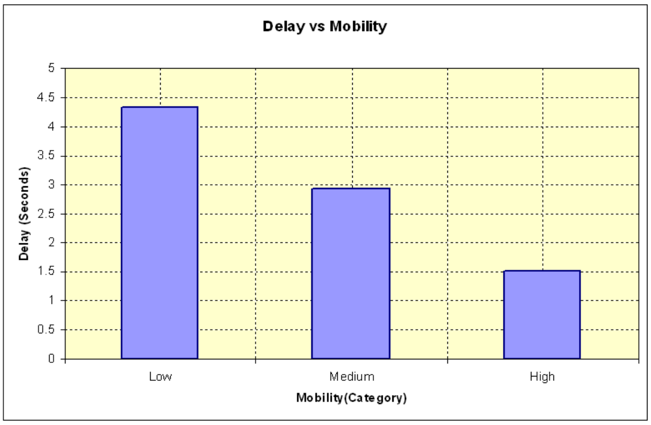


Fig. 4. Delay vs mobility

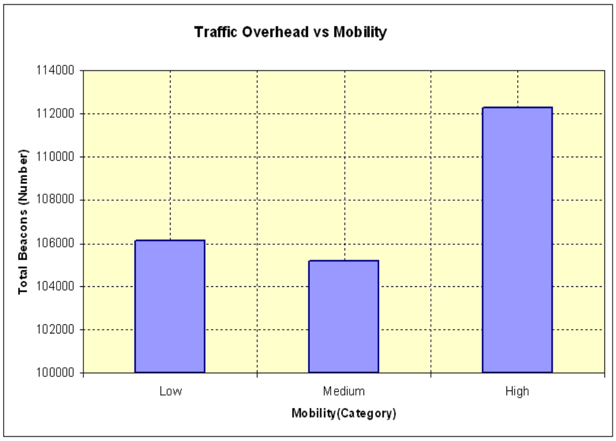


Fig. 5. Traffic overhead vs mobility

The performance of the framework with respect to the delay and traffic overhead were measured under conditions of low, medium and high mobility for 50 nodes offering 10 services over a 1000mX1000m square region. It can be observed from the results in Fig. 4, Fig. 5 that the framework performs well under conditions of high mobility. The delay in discovery of services is lesser because of an increase in the number of beacons transmitted due to the increased mobility. This increase results in an increased probability of the requesting node being in the range of a node having a suitable reply.

The average delay experienced by the MANET nodes during the startup phase of the framework *was comparable and even better than* existing optimal cross-layered solutions, which integrate service discovery with routing in an inflexible manner [17]. When comparing the delay experienced when using the framework to the delay experienced when using application layer service discovery protocols it was found that the speedup experienced *was greater than 60%* [18]. The overhead caused due to the operation in the link layer is estimated to be much less when compared to a similar operation in the application layer, as this would require transmission of lower layer PDUs. This overhead was found to be constant for a given number of nodes even though the number of services or service requests was increased. The framework's operation was analyzed immediately after the initialization of the framework to observe the worst case performance of the framework. Due to the caching employed by the components of the framework the operation of the framework is highly optimal once a significant amount of time has elapsed since the startup. The framework performs exceptionally well in MANETs that exhibit high mobility patterns. This is due to the fact that increased mobility implies increased probability of propagation of service information among the MANET nodes using the broadcast mechanisms.

One drawback of the framework implementation in the simulations is that because it employs the beacons to encapsulate its advertisements, during the service request/response the certainty of a service advertisement getting through is probabilistic at best [19]. However this can be overcome by simultaneous utilization of the link layer service discovery protocol and its application layer counterpart in replying to a service request. Also by ensuring a significant presence of SDs in the network this probabilistic nature of operation can be stabilized further.

6 Conclusion

Design of software for MANETs is different from the design for traditional wired networks because of the volatile and varying nature of such wireless networks. Typical challenges faced in the operation of MANETs are due to resource constraints, mobility and error proneness of wireless links. The exploitation of the opportunistic and unique modalities of communication offered by wireless communication will mitigate these drawbacks but the design of the protocol stack and applications that utilize it will need to incorporate novel enhancements for doing so. The framework for service discovery proposed in this research achieves this by modularization of different components involved in service discovery and localizing these components to different layers in protocol stack. This localization is done so as to benefit from the characteristics of operation in a particular layer that best suits the component. The framework takes an integrated and scalable approach to service discovery and access which can, if implemented on MANETs using diverse wireless technologies, facilitate widespread integration and adoption of wireless networks. This gives rise to plethora of interesting applications that revolves around the concept of interoperating mobile devices. The framework also takes on the challenge of operating in a resource constrained environment by optimizing the usage of resources in its operation without losing any of its flexibility.

References

1. Srivastava, V., Motani, M.: Cross-layer design: a survey and road ahead. In: IEEE Communi-cations Magazine, December 2005, vol. 43(12) (2005)
2. Kawadia, V., Kumar, P.R.: A cautionary perspective on cross-layer design. IEEE wireless communications (February 2005)
3. Jini Architecture Specification v1.2, Sun Microsystems (December 2001)
4. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol Version 2.0. IETF RFC 2608 (June 1999)
5. Chesire, S., Krochmal, M.: DNS-based Service Discovery. Internet Draft draft-cheshire-dnsext-dns-sd-04.txt, August 2006 (2007) (work in progress (expired) February 10, 2007)
6. UDDI specifications, Accessed on (August 1, 2006), `http://www.uddi.org/specification.html`
7. Goland, Y.Y., Cai, T., Leach, P., Gu, Y.: Simple Service Discovery Protocol 1.0. Internet Draft, work in progress (expired) (April 2000)
8. Helal, S., Desai, N., Verma, V., Lee, C.: Konark – A Service Discovery and Delivery Protocol for Ad-Hoc Networks. IEEE Wireless Communications and Net-working WCNC (2003)
9. Li, L., Lamont, L.: A Lightweight Service Discovery Mechanism for Mobile Ad Hoc Pervasive Environment Using Cross-layer Design. In: Communications Research Centre of Canada, PerCom Workshops (2005)

10. Koodli, R., Perkins, C.: Service Discovery in On-demand Ad Hoc Networks. Internet draft, work in progress (expired) (2002)
11. Cheng, L.: Service Advertisement and Discovery in Mobile Ad hoc Networks. In: Proc. of CSCW 2002, New Orleans, LA, USA (November 2002)
12. Chakraborty, D., Joshi, A., Yesha, Y.: Integrating Service Discovery with Routing and Session Management for Ad hoc Networks. Ad Hoc Networks Journal 4(2), 204–224 (2006)
13. Federal Information Processing Standards Publication 180-2, SHA-256, (August 1, 2002)
14. W3C Working Draft, XQuery 1.0: An XML query language (December 2001)
15. Sailhan, F., Issarny, V.: Scalable service discovery for MANET. In: Proceedings of the 3rd IEEE Int'l Conf. on Pervasive Computing and Communications (PerCom 2005) (2005)
16. The network simulator - ns-2, Accessed on (August 1, 2006), `http://www.isi.edu/nsnam/ns/`
17. Garcia-Macias, J.A., Torres, D.A.: Service discovery in mobile ad-hoc networks: Better at the network layer? In: Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW 2005) (2005)
18. Liu, H., Barbeau, M.: Performance evaluation of service discovery strategies in ad hoc networks. In: Proceedings of the second annual conference on communication networks and services research (CSNR 2004) (2004)
19. Huang, L., Lai, T.-H.: On the scalability of IEEE 802.11 ad hoc networks. MOBI-HOC 2002, EPFL Lausanne, Switzerland (June 2002)

Cross-Layer Designs Architecture for LEO Satellite Ad Hoc Network

Zhijiang Chang and Georgi Gaydadjiev

Computer Engineering Laboratory
Delft University of Technology
Mekelweg 4, 2628 CD Delft, the Netherlands
Telephone:. +31 15 278 6177
{zhijiangchang,georgi}@ce.et.tudelft.nl

Abstract. Future Low Earth Orbit (LEO) satellite networks are envisioned as distributed architectures of autonomous data processing nodes. Such ad hoc networks should deliver reliable communication channels for control commands and data among ground stations and satellites minimizing delay and power. The LEO satellite networks are different from the generic ad hoc scenario. In this paper, we first analyze the specifics of ad hoc LEO satellite networks. Next, we propose a cross-layer protocol architecture that includes three cross-layer optimizations: simple integrated MAC/PHY layer, novel Balanced Predictable Routing (BPR) and a dedicated QoS aware TCP sliding window control mechanism. They all contribute to the end-to-end delays improvement and successful delivery increase. It also fulfills the QoS requirements. According to our simulations, the coverage of ground stations is improved. The throughput percentage of all data types is improved by 5.8% on average and the QoS of high priority application is guaranteed.

Keywords: cross-layer design, LEO satellite network, ad hoc network.

1 Introduction

The future satellite applications require self-organized, dynamic network topology without predefined constellation [1,2]. Such applications include: 1) deep-space exploration that uses the Inter-Satellite Links (ISLs) of LEO network to communicate to the control center; 2) LEO satellites control that accesses satellites that do not have direct link to Ground Station (GS); 3) satellite telephone service in which the ground terminal is not directly connected to the satellite networks with a constellation. In order to support the above applications, the future satellite networks should be able to maintain the connectivity and efficiency when satellites join or leave the networks dynamically. Such networks are ad hoc networks. That will be characterized by their frequently changing topology and intermittent connectivity.

The current LEO micro satellites and their networking, however, do not meet the requirements to support the future applications. First, the satellites are launched by many different organizations for various purposes. Their efficiency is extremely low because most of them work alone. This is because: 1) the standalone LEO satellites are visible to GS control only for 10 to 20 minutes per rotation; 2) the LEO satellite's average lifetime is only around 6 years [3]. Secondly, the heterogeneous satellites have various computing and power capacities. This limits not only the efficiency of individual satellite, but also the potential cooperation among them. The heterogeneous architectures also lead to more complicated network organization, faster changing topology and less stable communication channels.

In order to develop a satellite networking system that fulfills the requirements of future applications and overcomes the above problems, we take the advantages of cross-layer designs. The cross-layer designs optimize the overall network performance by scarifying the layers' interdependencies [4]. A strict modularity and layer independence may lead to non-optimal performance in the future satellite networks. The heterogeneous network requires adaptability provided by the cross-layer designs. Furthermore, the LEO satellites can use cross-layer information such as signal strength variation to predict the motion of other satellites and the communication link quality.

The main contributions of this paper are:

– Analysis of QoS requirements and the related energy efficiency for ad hoc satellite networking context;
– Integrated MAC/PHY layer that provides network congestion and link quality information;
– Novel Balanced Predictable Routing (BPR) based on Dynamic Source Routing (DSR);
– A QoS aware TCP congestion control algorithm that especially ensures the delivery of the control commands;
– Careful simulation that validates our cross-layer architecture using ns-2.

This paper is organized as follows. We outline the special QoS requirements and problems of LEO satellite networks in section 2. Related work is discussed in section 5. Section 3 presents the proposed architecture and the three cross-layer optimizations. The simulation results are discussed in section 4. We finally conclude the paper in section 6.

2 Special Issues of LEO Satellite Networking

Some special issues of the LEO satellite ad hoc network are considered while we design the cross-layer architecture. First, the satellite networks have special runtime QoS requirements for mission control and payload specific onboard applications. Second, different link quality of different types of ISLs is essential for the satellite networks. This section discusses these special issues that have significant impact on the proposed cross-layer architecture.

2.1 QoS Requirements for Applications

The satellite networks have strict bandwidth limitations. Different types of application in the LEO satellite networks have different bandwidth requirements for QoS. We classify the traffic flows to three classes: (i) the mission control flows that should be delivered at all costs are given the highest priority; (ii) the real-time services such as satellite telephone that have bandwidth and delay requirements have the second high priority; (iii) the non-realtime services such as FTP that can be delivered at best effort (as good as possible) have the lowest priority.

2.2 Inherit Problems of LEO Satellite Networks

Providing full ad hoc connectivity in the LEO satellite network is a challenging task. The following problems in the satellite networks should be considered: (i) High Bit Error Rates (BER) on satellite links are caused by signal interferences, such as atmospheric or ionosphere effects and artificial jamming. For instance, the quality of ISL changes rapidly when the link path goes through the atmosphere. (ii) Load balancing is a major problem in such networks because of the limited power budget of nodes. The normal routing protocols are more likely to use links with better quality such as shorter delay than links with longer delay. The traffic load should be fairly distributed in the network in order to avoid exhausting some satellites' energy when leaving others unused. (iv) Different types of ISLs have strong impact on the network topology. Intra-plane ISLs and Inter-plane ISLs of adjacent planes are stable links because that the relative position of the satellites is stable for certain periods. Cross-seam ISLs are fast changing unstable links that are only available to a short period.

3 Cross-Layer Architecture For LEO Satellite Networks

We propose an cross-layer architecture that involves three cross-layer designs. These three designs map QoS control at all layers, all being time-varying. The cross-layer optimizations provide not only QoS control to the applications, but also approaches to overcome the inherit problems of the satellite networks as stated in the previous section. The first optimization is an integrated MAC/PHY layer that provides more accurate and adequate information to other cross-layer optimizations. The second optimization controls the sliding window of TCP protocol in order to guarantee the delivery of application data with higher priority. The third optimization adapts the Dynamic Source Routing (DSR) protocol to LEO satellite network to use more stable links and balance the traffic in the network at the same time. Multiple cross-layer designs have potential risk of malfunctioning when interacting with each other. Such problems includes shared information access and adaptation loops [5]. In order to prevent such problems, we use the infrastructure for cross-layer design interaction proposed in [6] to ensure that the three optimizations are loop-free, and behave correctly according to their designs.

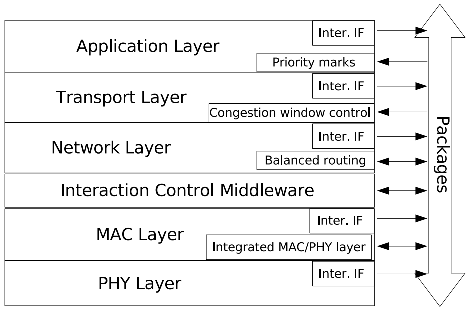


Fig. 1. The cross-layer satellite architecture

As shown in figure 1, the following cross-layer information is propagated in the protocol stack: the priority information provided by the applications; the wireless link quality information by the integrated MAC/PHY layer. The routing protocol uses links with better quality using the wireless link quality information. The TCP layer adjusts the congestion window size according to the MAC/PHY layer information and the application priority.

3.1 Integrated MAC and PHY Layer

We propose an integrated MAC/PHY layer to simplify the information provided to the upper layers. We propose a normalized variable called Degree of Collision (DOC) to represent a node's wireless link quality. The two main contributions of the paper are: 1) the Balanced Predictable Routing (BPR) mechanism presented in section 3.2 and 2) the QoS aware TCP sliding window control introduced in section 3.3. We only use DOC as an attribute representing the ranking of link quality in the BPR and QoS aware TCP control. The performance of BPR and QoS aware TCP sliding window control are invariant to how the DOC is calculated. We do not focus on optimal DOC calculation, but we rather provide a simple straight-forward expression as a proof of concept. For instance, if we calculate P_e using both the distance and the satellite's angular position to ground station as parameters, the DOC can represent the link quality more accurately. But DOC accuracy does not influence the performance of the proposed TCP sliding window algorithm and BPR, therefore DOC optimizations are outside the scope of this paper.

DOC is calculated using two probability functions: 1) *error probability* P_e as a function of BER and SNR, and 2) *collision possibility* P_c of outgoing packets from this node. A higher DOC value indicates more network congestion or package loss (1).

$$DOC = f(P_e, P_c), DOC \in (0, 1] \tag{1}$$

The errors related to noise and collision are equally important indications of link quality. Therefore the DOC is calculated as the weighted sum of probability of error P_e and collision P_c (2). For simplification, we use $W_e = W_c = 0.5$ in our simulation. A more careful selection of the two parameters may improve the accuracy of DOC. But as stated earlier, this accuracy does not influence the performance of the proposed mechanisms.

$$DOC = W_e \times P_e + W_c \times P_c \tag{2}$$

$P_c = N_c/N$ is the ratio of collided packets N_c and total packets in an observation period N. In Space, where we can safely assume that no obstacle stands on the path between the satellites, the distance is the dominating element of the error probability. Therefore, P_e is calculated using exponential distribution probability density function with the distance as a parameter:

$$P_e = f(x;\lambda) = \lambda \times e^{-\lambda x},\ \lambda = 1 \text{ and } x = Range_{max} - distance \tag{3}$$

In (3), $Range_{max}$ is the satellite's maximum communication range (distance to GS) in the order of 10^4 meters. This simulates that the error probability sharply increases when the distance approaches the maximum range (in the last 30 kilometers). P_e equals to 1 when the distance is larger than $Range_{max}$. The range of ISL can hardly exceed the distance between the GS and the satellite because the ground stations have much higher receiver gain than the satellites.

3.2 Balanced Predictable Routing

We propose a Balanced Predictable Routing (BPR) that emphasizes cross-node cooperation and cross-layer optimization within an individual node. First, the BPR uses predicted stability of a route to guarantee successful delivery. Second, the BPR provides load balancing for the entire network. Without a load balancing mechanism, the more stable routes are expected to be overloaded because the stability mechanism always selects them.

There are many ad hoc routing protocols such as Dynamic Source routing (DSR) [7], Ad-hoc On-demand Distance Vector (AODV) [8] and Temporally-Ordered Routing Algorithm (TORA) [9]. We develop the BPR based on DSR for the following reasons: (i)DSR is on-demand routing that does not use periodic messages to update the routing information. Consequently, it consumes less bandwidth and energy than table-driven (proactive) routing protocols. According to [10], DSR has smaller routing overhead than other protocols when the nodes never pause like the satellites. (ii) DSR records the complete route from source to destination. Therefore, the source node can optimize the route using all the intermediate links' information. (iii) The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead. (iv) DSR does not maintain a routing table and consequently needs less memory space. (v) The LEO satellite network has limited hop-count (from one to three in our simulation). A simple node identifier instead of full IP address can be used in satellite networks. Both the limited hop-count and the simple identifier reduce the overhead in packet headers, which is the main disadvantage of the DSR.

In order to rank the routes to the same destination according to the link stability, we add a variable $S \in (0, 1]$ in the routing cache to indicate the stability of the route as shown in the algorithm below. The value of S equals to 1 when the route is most stable. The value of S decreases when the route becomes less stable. S is periodically updated during the time when the satellite travels between the two polar regions. More frequent update makes the link information more up-to-date, but leads to more computation overhead. We update S 50 times in our simulation. The value of S is calculated according to DOC and the availability of the route. If the satellite passes the polar region and starts moving in another direction, S is reset to 1 and the calculation starts over. The following pseudocode presents the algorithm to calculate the stability variable.

```
WHILE {traveling from one polar to the other}
    S = 1
    FOR {each observation time}
        IF {if route is available}
            S = S / (1 + DOC)
        ELSE {route is broken}
            S = S/2
        ENDIF
    ENDFOR
ENDWHILE
```

The stability variable is being constantly calculated when the satellite travels between the polar regions. The variable is reset to 1 in two polar region. The above actions are taken because: (i) some Micro LEO satellites go into standby mode, or even power off inside the polar region; (ii) the distance between satellites rapidly changes in the polar region and many entries in the routing table need to be recalculated; (iii) even without route entry reconstruction, the satellites may still be overloaded due to massive possible handoffs.

We use the link quality variable DOC instead of orbit information to predict future condition of routes because (i) the LEO satellites with different power and antenna capacity do not necessarily share a good communication channel even when they are close; (ii) the Global Positioning System (GPS) is not available on most micro LEO satellites.

With the stability variable, the BPR behaves differently from the original DSR in the following aspects: (i) The stability variable S is broadcasted along with the routing information in the route discovery package during the route discovery phase. (ii) Unlike the DSR that only stores one route to destination in the route cache, BPR stores multiple routes to the same destination. The route with highest stability S is selected in the route discovery phase. (iii) If a node considers itself overloaded, it drops the Route Request message in the route discovery phase. (iv)The reverse route is used to return the Route Reply message. The stability variable S is not attached to the Route Reply message. (v)All nodes overhear the broadcasted Route Request messages to update the stability variable of each route. (vi) In the Route Maintenance Phase, the erroneous hop is not removed from the node's route cache. Instead, the stability variable S of all routes containing the hop is updated. (vii) When the satellites pass the polar region, the route cache is cleaned. Therefore, all the routes are recalculated on-demand.

Considering the antenna and computing capacity of LEO satellites, we propose to use the stable intra-plane and inter-plane ISLs, when avoiding the fast changing cross-seam links. The proposed algorithm distinguishes well the more stable ISLs from the cross-seam ISLs and other fast changing ISLs, because the fast changing links always have smaller S. If the link quality of an ISL varies in a short range during observation period, it is going to maintain the quality level in the future. This is because the satellites move on orbits so that the satellite's relative position varies in a small range even when the satellites' absolute speed is very high. Eventually, such satellites can develop a constellation without orbit information of each other.

In order to achieve load balancing, we use the local and global view[1] [11] that is already provided by the cross-layer interaction architecture [6]. The node is overloaded if the ratio of local and global views is greater than 1 [11]. Remaining energy is also an important indications of the overload. Therefore, the satellites running out of power are considered overloaded.

3.3 QoS Aware TCP Congestion Control

The general approach to guarantee the delivery of high priority packets is the prioritized queuing at the IP or MAC layers. This approach, however, rearranges the order of the outgoing packets only. In a wireless network, the interference increases when the number of packets in media increases. This means the number of packets sent to the wireless media should also be controlled in order to reduce interference. Consequently, we propose a QoS aware TCP congestion control mechanism to reduce the number of outing packets when the high priority packets needs to be sent.

Our proposal dynamically controls the TCP sliding window size to reduce the wireless media interference and delay of high priority application such as the control command. According to the standard sliding window control algorithm, the window size is half of the original value when congestion happens in the stable phase. We borrow the idea to temporary reduce the network traffic for a very short time. The QoS aware sliding window control mechanism reduces the window size during the time when the satellite node is sending or relaying the command flow. The window size should be quickly reduced when link quality is bad or collision probability is high. On the other hand, if the link quality is high and collision probability is low, the window should be slowly reduced. Therefore, we use formula 4 to adjust the congestion window size.

$$Size_{window} = \frac{Size_{window}}{(1 + DOC^{e})}, e = NapierConstant \tag{4}$$

The sliding window size is reverse proportional to DOC. According to formula 4, the sliding window is divided by a value in the range between 1 and 2.

[1] The local view presents the queuing length of the node; The global view presents the average queuing length of the neighboring nodes.

We use the Napier constant e in DOC^e as shown in (4) so that the overall throughput of the network is not sharply reduced if the network has little congestion or other kinds of packet loss. In other words, a small DOC does not affect the behavior of the TCP sliding window.

4 Validation and Results

In order to validate our cross-layer optimizations, we implement the cross-layer optimizations in the Network Simulator 2 (ns-2) version 2.28 [12]. Our simulation is based on the ns-2 satellite package provided by [13]. We made the following improvements to the satellite package: (i) The energy model is introduced to simulate the satellite's behavior without the energy source (in the shadow of the earth); (ii) 802.11 MAC like collision model is introduced to calculate the collision probability in satellite network; (iii) The centralized routing of satellite package is replaced by BPR.

4.1 Simulation Scenario

We use the following configuration in our simulation: 1 to 17 Satellite nodes on random polar orbits (altitude 500-800km) within 5 degrees deviation with random start elevation degree (based on longitude 4.0 E); GSL from 500kbps to 2Mbps for each satellite; ISL from 1Mbps to 2Mbps; both symmetric links; Two Ground stations: A (in Delft 51.9792 N, 4.375 E): B (New York 40.30N, 73.24W); Data Sources: 10 CBR on UDP simulates the realtime data from GS A to B, 20 FTP on TCP simulates non-realtime data from satellites to GS, FTP to simulate control commands sent from ground stations to satellites when there is a connection; duration: 1 day (86400s).

4.2 Performance Analysis

Assuming T_i is the time when satellite i is connected to GS and T_{total} is the total fly time of all satellites, the coverage is defined as $\frac{\sum T_i}{T_{total}}$. As shown in figure 2, the coverage improvement mainly happens when there are 5 to 11 satellites in the formation. In this case, the margin effect of the elevation mask has a strong impact on the links' availability. The margin effect is caused by interference from atmosphere and the relative position between the satellite and satellite/GS. When the satellites start moving into or leaving the elevation mask, the link quality changes very rapidly. The cross-layer design using MAC and PHY layer feedbacks can predict the satellites that are falling out of line of sight and consequently switch to other satellites for communication in advance. The gain of cross-layer optimization decreases when there are more satellites in the network. This is because multiple routes are available when multiple ISLs are above the elevation mask. Consequently the margin effect is avoided by switching to other links.

We compare the successful delivery in term of throughput percentage w/o cross-layer designs. Both the individual cross-layer designs and their combination

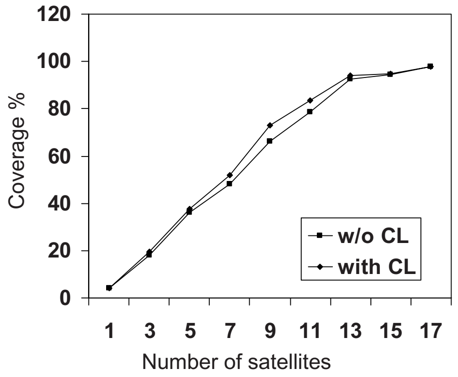


Fig. 2. coverage comparison with and without CL design

are simulated. In figure 3, "CL 1" is the integrated MAC/PHY layer; "CL 2" is the BPR; and "CL 3" is QoS aware TCP sliding window control. The throughput percentage is improved when we use the integration of the three optimizations because (i) the BPR distributes the traffic more evenly in the network, which leads to less congestion on route with better link quality; (ii) the stability variable algorithm ensures that fast changing ISLs and satellites in energy conservation mode are avoided in the route.

The cross-layer design interaction architecture insures that the system benefits from all cross-layer designs. In figure 3, the "CL 3" is important to improve the delivery of command data then the network has higher traffic load (13 to 17 nodes). This optimization has little impact on the realtime and non-realtime traffics when the network load is low. This validates our design that it should not affect low priority services when the network is not congested. The result in figure 3 also shows that the delivery percentage is lowest when the constellation consists of 7 satellites. This is because the single path, however, is unstable due to the margin effect of the elevation mask and the fast changing distance. When the number of satellites increases, multiple routes are available at the same time. Consequently, the delivery failure is reduced.

The delivery percentage in figure 3 is high ($\geq$ 88%) because the satellite-GS link is in optimized status. And due to bandwidth constraints, the TCP sliding window is always small, which also leads to reduced network congestion. This, however, is not true in real environment. In the real environment, the ISLs can be asymmetric links with various bandwidth, which leads to significant packet lost. This cannot be simulated in the ns-2 simulator.

Table 1 compares the average number of forwarded packets per satellite and its standard deviation. The results show that the cross-layer architecture not only improves the throughput, but also distributes the traffic more evenly in the network. Considering that most of satellite's energy is consumed by the telecommunication system, we reduce the chance of exhausting some satellites while leaving others full of energy.

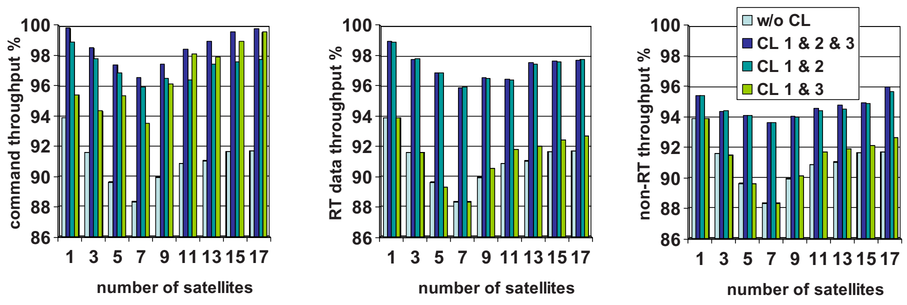


Fig. 3. The throughput % of prioritized traffics

Table 1. Average and standard deviation of number of forwarded packets per satellite

sat.	5	7	9	11	13	15	17
w/o CL average	13577.4	17497.3	19257.6	20431.9	20954.8	21314.5	21853.6
w/o CL deviation	1185.1	953.7	2249.7	1835.3	1973.8	1745.3	2034.6
with CL average	14671.2	18616.7	20266.2	21557.2	21574.6	22054.8	22612.9
with CL deviation	232.0	516.1	1024.0	966.5	1054.3	1234.7	1095.3

Our proposal also improves the end-to-end delay of packets as shown in figure 4 because the MAC layer retransmission is reduced by using more stable links. Our proposal, however, may have negative effect on the delay for the following reasons. First, the load balancing algorithm pushes some packets to the edge of the network in order to reduce traffic in its center. This action increases the hop count of those packets. Second, the BPR always prefers to use the more stable links, which increases the queuing length of nodes with good link quality.

4.3 Overhead Analysis

The proposed cross-layer designs introduce internal overhead within a node as well as external overhead on the network. The internal and external overhead is calculated as the number of bytes added to normal packets to carry cross-layer information. The internal overhead consists of two parts: the overhead of the individual cross-layer design and the overhead of the architecture to enable correct interaction among multiple designs. The three cross-layer designs and the interaction architecture together introduce an internal overhead less than 0.25%. Therefore, the internal overhead is neglectable. In our proposal, the external overhead is the one-hop neighbors' information used to calculate the global view in order to achieve load balancing. Because the satellites have very limited one-hop neighbors (between one and three in our simulation) the external overhead is low (below 2%). Figure 5 depicts the internal and external overheads of our proposal.

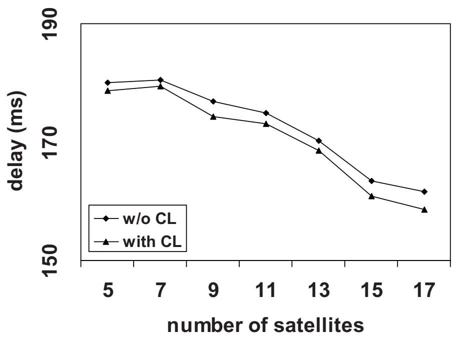


Fig. 4. The average packet delay from GS A to GS B

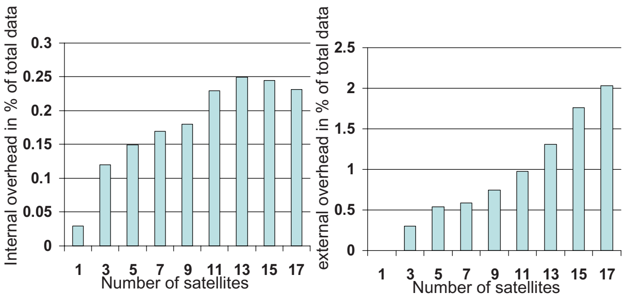


Fig. 5. The internal and external overheads of cross-layer designs

5 Related Works

Previous research shows cross-layer designs are promising for QoS control in the MANET. In [14], a cross-layer framework for WLAN QoS support is proposed. The authors show that QoS at MAC layer can be optimized by taking advantage from the IP, TCP and application layers. In [15], a combined cross-layer design for QoS content delivery is proposed. The authors introduced a QoS-aware scheduler and power adaptation scheme at the MAC layer for an efficient resource utilization in the upper layers. Their results show that the cross-layer design provides a good scheme for wireless QoS content delivery. These works can not be directly compared to our proposal because our proposal considers the characteristics and special requirements of the satellite networks instead of MANET. The above works provide the general approach of QoS optimization that involves all layers in the protocol stack.

Previous works also focus on MANET higher layer optimization using information from MAC layer and below. In the traditional wired communication world, the bit error rate (BER) of the link can be neglected. The TCP layer assumes that the package loss is an indication of congestion. In the wireless world, however, the package loss is mainly caused by loss on the wireless link instead of congestion. Many papers have analyzed this problem and proposed solutions such as using link connectivity to notify the TCP protocol if congestion really happens [16] and [17]. Vania Conan et al. proposed in [18] the WIDENS architecture. This architecture emphasizes the low-level protocol integration by virtually providing a super low layer that combines the DLC, MAC and PHY layers. Special communication channel between the network layer and integrated low layer is also established in order to support hard QoS routing in Mobile Ad hoc NETwork (MANET). In [11], Rolf Winter et al. proposed the CrossTalk architecture. Unlike the above designs, the CrossTalk architecture emphasizes cross-node cooperation as well as cross-layer design within individual node. The above works provide guidance for the proposed BPR and QoS aware TCP sliding window control mechanism.

6 Conclusion and Future Work

Future LEO satellite networks are expected to have a dynamic topology and become ad hoc networks. The current architecture of the LEO satellite networks, however, cannot fulfill the requirements for such an fast changing network environment. In this paper, we first discussed the special QoS requirements and the inherit problems of the LEO satellite networks. Then, we proposed two cross-layer designs, namely BPR and QoS aware TCP sliding window control, both using information from an integrated MAC/PHY layer. The BPR improved the total throughput while considering the load balancing at the same time. The QoS aware TCP mechanism guaranteed the delivery of high priority services while avoiding unnecessary decrease of the total throughput. The end-to-end delay was also reduced because BPR reduced the MAC layer retransmission by selecting the links with better quality. In the future, we will continue designing and simulating the cross-layer optimizations such as the energy consumption control in the LEO satellite network environment. This will provide more systematic solutions for the future satellite networks.

Acknowledgments. This work is supported by the Dutch Government as part of the decree on subsidies for investments in the knowledge infrastructure (Bsik) program. This work is done within the Micro Satellite (MISAT) project.

References

1. Prescott, G., Smith, S., Moe, K.: Real-time information system technology challenges for nasa's earth science enterprise. In: 20th IEEE Real-Time Systems Symposium, Phoenix, AZ., U.S.A. (December 1999)
2. Shen, C.C., Rajagopalan, S., Borkar, G., Jaikaeo, C.: A flexible routing architecture for ad hoc space networks. Computer Networks 46(3), 389–410 (2004)
3. Giambene, G., Chini, P.: Introduction to satelliste communications and resource management. In: chapter of Resource Management in Satellite Networks: Optimization and Cross-Layer Design (March 2007)

4. Kawadia, V., Kumar, P.R.: A cautionary perspective on cross layer design. In: IEEE Wireless Commun., February 2005, vol. 12(1), pp. 3–11. IEEE Computer Society Press, Los Alamitos (2005)
5. Srivastava, V., Motani, M.: The road ahead for cross-layer design. In: Proceedings of 2005 2nd International Conference on Broadband Networks, pp. 551–556. IEEE, Los Alamitos (2005)
6. Chang, Z., Gaydadjiev, G.N., Vassiliadis, S.: Infrastructure for cross-layer designs interaction. In: the 16th IEEE International Conference on Computer Communications and Networks (IC3N), August 2007, pp. 19–25 (2007)
7. Johnson, D.B., Maltz, D.A., Hu, Y.C., Jetcheva, J.: The Dynamic Source Routing protocol for mobile ad hoc networks (dsr). In: IETF Draft (April 2003)
8. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (aodv) routing. In: Internet Engineering Task Force (IETF) draft (July 2003)
9. Park, V.D., Corson, M.S.: Temporally-ordered routing algorithm (tora) version 1: functional specification. In: Internet Engineering Task Force (IETF) draft (November 1997)
10. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocols. In: Mobile Computing and Networking (MobiCom), pp. 85–97 (1998)
11. Winter, R., Schiller, J.H., Nikaein, N., Bonnet, C.: Crosstalk: cross-layer decision support based on global knowledge. In: Communications Magazine, vol. 44(1), pp. 93–99. IEEE, Los Alamitos (2006)
12. ns 2: The network simulator version 2, http://www.isi.edu/nsnam/ns/
13. Henderson, T.R., Katz, R.H.: Network simulation for leo satellite networks. In: American Institute of Aeronautics and Astronautics (2000)
14. Pau, G., Maniezzo, D., Das, S., Lim, Y., Pyon, J., Yu, H., Gerla, M.: Cross-layer framework for wireless lan qos support. In: the IEEE International Conference on Information Technology Research and Education (ITRE) (2003)
15. Chen, J., Lv, T., Zheng, H.: Joint cross-layer design for wireless qos content delivery. In: IEEE International Conference on Communication (2004)
16. Raisinghani, V.T., Singh, A.K., Iyer, S.: Improving tcp performance over mobile wireless environments using cross layer feedback. In: IEEE International Conference on Personal Wireless Communications, New Delhi, India (2002)
17. Shakkottai, S., Rappaport, T.S., Karlsson, P.C.: Cross layer design for wireless networks. In: IEEE Commun. Mag., October 2003, pp. 74–80. IEEE, Los Alamitos (2003)
18. Aiache, H., Conan, V., Barcelo, J.M., etc., L.C.: Widens: Advanced wireless ad-hoc networks for public safety. IEEE Computer Society Press, Los Alamitos (2005)

MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks

Gerald Wagenknecht, Markus Anwander, Torsten Braun, Thomas Staub, James Matheka, and Simon Morgenthaler

Institute of Computer Science and Applied Mathematics
University of Bern, Switzerland
{wagen,anwander,braun,staub,matheka,morgenthaler}@iam.unibe.ch

Abstract. In this paper we present a new management architecture for heterogeneous wireless sensor networks (WSNs) called MARWIS. It supports common management tasks such as monitoring, (re)configuration, and updating program code in a WSN and considers specific characteristics of WSNs and restricted physical resources of the nodes such as battery, computing power, memory or network bandwidth and link quality. To handle large heterogeneous WSN we propose to subdivide it into smaller sensor subnetworks (SSNs), which contains sensor node of one type. A wireless mesh network (WMN) operates as backbone and builds the communication gateway between these SSNs. We show that the packet loss and the round trip time are decreased significantly in such an architecture. The mesh nodes operate also as a communication gateway between the different SSNs and perform the management tasks. All management tasks are controlled by a management station located in the Internet.

1 Introduction

A heterogeneous wireless sensor network (WSN) consists of several different types of sensor nodes (SNs). Various applications supporting different tasks, e.g., event detection, localization, and monitoring may run on these specialized SNs. In addition, new applications have to be deployed as well as new configurations and bug fixes have to be applied during the lifetime. In a network with thousands of nodes, this is a very complex task and a general management architecture is required. The questions are, how we can achieve that the monitoring, the configuration, and the code updating can be performed on heterogeneous sensor nodes during their life-time? How has such a heterogeneous WSN to be structured to handle these management tasks efficiently and automatically over the network?

In this paper, the usage of a wireless mesh network (WMN) as a backbone to build a heterogeneous WSN is motivated. The proposed new management architecture called MARWIS supports common management tasks such as monitoring the WSN, configuration of the WSN, and code updates.

This paper is structured as follows: Section 2 introduces related work on management of WSNs, middleware, code distribution, and reprogramming. Afterwards, the management architecture MARWIS is presented, including the description of a heterogeneous WSN using a WMN backbone (Section 3), the specification of the infrastructural elements (Section 4), and the management protocols (Section 5). The implementation of MARWIS is described in Section 6, the evaluation of the advantages of using WMNs as backbone for heterogenous WSNs in Section 7. A conclusion is presented in Section 8.

2 Related Work

Most management and code distribution approaches does not support heterogeneous WSN environments and distribute specific code for such SN platforms. However, advanced WSNs are typically composed of rather heterogeneous SNs, since the functionality required is highly versatile. To improve current research, our concept adds mechanisms to support heterogeneity for management in WSNs. In [1], we presented a short overview of the management architecture to be described in much more detail in this paper.

MANNA [2] is a management architecture for WSNs. It provides functions to establish configurations for WSN entities. The deployment of several manager nodes in a hierarchical way based on clustering has been proposed. TinyCubus [3] is a management and configuration framework for WSNs. It is based on a clustered architecture and assigns certain roles to the SNs. Another focus of TinyCubus is code distribution, minimizing the code fragments to be distributed in a WSN. The so-called Guerrilla management architecture [4] facilitates adaptive and autonomous management of heterogeneous ad hoc networks.

Promising concepts to hide hardware heterogeneity in WSNs are middleware approaches as presented in [5] based on either scripting language interpreters or virtual machines. MiLAN [6] provides a set of middleware mechanisms for adapting the WSN to affect the application supplied performance policy. No support for dynamic code update is included as its operation may not be changed at run-time. Impala [7] is a middleware architecture that enables modular application updates and offers repair capabilities for WSNs Maté [8] is a byte-code interpreter (virtual machine) running on TinyOS and allows run-time reprogramming. The Global Sensor Network (GSN) [9] provides a middleware for fast and flexible integration and deployment of heterogeneous WSN.

Surveys of software update techniques in WSNs are presented in [10], [11] and [12]. They focus on the execution environments at the SNs, the software distribution protocols in the network and optimization of transmitted updates. The authors of [13] propose efficient code distribution in WSNs. The focus is the reduction of the total amount of data for a code update by only transmitting the differences between the old and new code. Different optimizations like address shifts, padding and address patching are made. In [14] an incremental network programming protocol, which uses the Rsync algorithm to find variable-sized blocks that exist in both code images and then only transmits the differences is presented. In [15] a scheme that uses incremental linking to reduce the number of changes in the code and transmits the code update with a diff-like algorithm is described. FlexCup [16] is more flexible, as the linking process is not performed at the base station, but on the SNs. Multi-hop Over-the-Air Programming (MOAP) [17] is a code distribution mechanism specifically targeted for Mica-2 motes. It focuses on energy-efficient and reliable code distribution.

3 WSN Management Scenario and Tasks

Many different applications may run in a WSN, e.g., event detection, localization, tracking, monitoring. Therefore, different types of SNs, which might measure different sensor values and perform different tasks, are required. Existing SN platforms in general have different radio modules, which are not interoperable. SNs of the same type build a sensor subnetwork (SSN), which is not able to communicate directly to another SSN. A heterogeneous WSN is built from several SSNs. To interconnect such a heterogeneous WSN mesh nodes (MNs) are proposed as gateways between these SSNs. A SN plugged into a serial interface (e.g. USB) to a MN works as gateway. The wireless MNs communicate among each other via IEEE 802.11. A possible scenario is shown in Fig. 1.

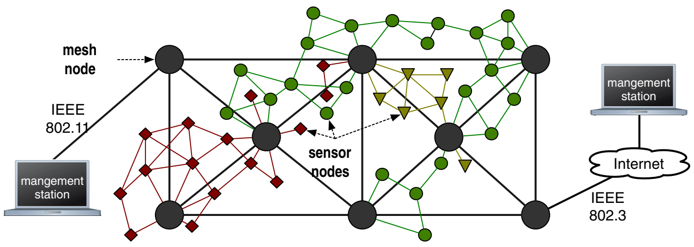


Fig. 1. A possible scenario for heterogeneous WSNs with management devices

Currently available sensor nodes are mainly prototypes for research purposes. We have evaluated a number of sensor nodes and selected four of them to build a heterogeneous WSN: ESB nodes [18], tmote SKY [19], BTnodes [20], and MICAz [21]. For the management backbone a WMN consisting of mesh nodes with two IEEE 802.11g interfaces, an AMD Geode 233 CPU and 128 MB RAM have been selected. A 8 GB CompactFlash card can be attached.

The use of such an architecture with a WMN as backbone has various advantages. In addition to the communication gateway functions MNs further perform management tasks for heterogeneous WSNs. The main benefit is the ability to communicate with different types of SNs in several SSNs. Moreover, the use of a WMN has advantages by subdividing a huge WSN into smaller SSNs. A WSN consisting of thousands of nodes and one base station creates many communication problems. Most of them are caused by the high number of SN hops in larger WSNs. Subdivision into smaller SSNs limits the sensitive SN links to 3 or 4 hops to the next sensor node gateway. These results in a better communication performance with a clearly lower packet delay, jitter and packet loss. SNs in the vicinity of the sink preserve energy and processing power, because they do not have to forward the whole WSN traffic. Another advantage of using a WMN is that a new SN platform can be easily inserted into the heterogeneous WSN by plugging a SN gateway into a MN. The IP address allocation depends on the corresponding MN, which makes it possible to allocate similar IP address in a physical neighborhood.

The MNs also provide management functionalities for heterogeneous WSNs. Hence, the limited SNs have less management functions to perform, which decreases memory and computation requirements. In a heterogeneous WSN with a large number of different SNs, a comprehensive management architecture is required. In addition to the MNs, providing the management functionality, there are one or more management stations (see Fig. 1). A user performs the management tasks with their support. From the management point of view there are several tasks required to manage a heterogeneous WSN. In general, the tasks can be divided into four areas: (1) monitoring the WSN and the SNs,(2) (re)configuring the WSN and the SNs, and (3) updating and reprogramming the SNs.

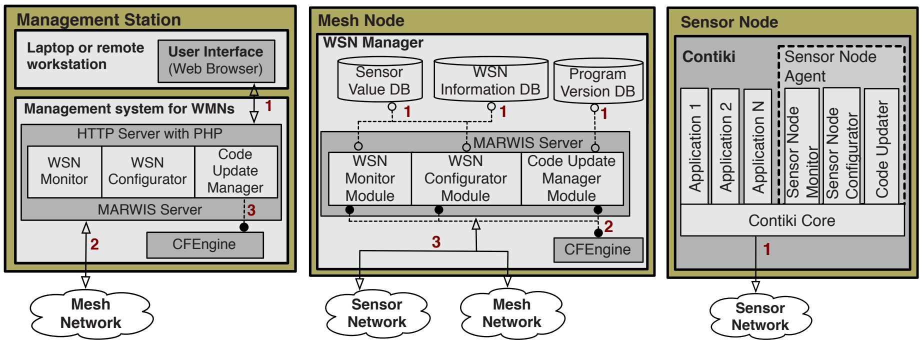
The management tasks include visualization of all SNs in the several subnetworks at the management station. Furthermore, status information about the SNs has to be monitored and displayed. This includes SN hardware features (micro-controller, memory, transceiver), SN software details (operating system versions, protocols, applications), dynamic properties (battery, free memory), and, if available, position information. SN configuration includes configuring the SNs, the running applications or the network. Updating and reprogramming the SNs is a very important issue. In a large WSN manual execution of this task is not feasible. A mechanism to handle this automatically and dynamically over the network is required. Both the operating system and applications must be updated, fully or partially. Mechanisms to handle incomplete, inconsistent, and failed updates have to be provided. Aggregating and managing the sensor values includes mechanisms to store and download the collected data from the SNs. There are many existing mechanisms and protocols, which have been designed for aggregating data from SNs (e.g. Directed Diffusion [25]). Therefore, in our architecture this task is treated as optional and not as major issue.

4 Management Architecture

The architecture to manage heterogeneous WSNs efficiently contains the following structural elements: one or more management stations, several MNs as management nodes, SN gateways plugged into a MN, and the different SNs. These elements are shown in (Fig. 2).

4.1 Management Station with Management System for WMNs

The **management station.** is divided into two parts. It consists of a laptop or remote workstation to access a graphical user interface to control the WSN and a management system for WMNs [22], which is connected to the Internet and can be accessed by the remote workstation from anywhere. It includes a web server and is shown in Fig. 2(a).



(a) Management station (b) Mesh node (c) Sensor node

Fig. 2. Architecture of the MARWIS elements

The **user interface** displays the WSN topology with the MNs including the subordinated SNs and information about the SNs (**1** in Fig. 2(a)). The **management system for WMNs** contains a small Linux distribution [22] including all required applications, especially a HTTPS server, which maintains different modules to handle the requests and transmits them to MNs, SNs or CFEngine [23], such as **WSN monitor**, **WSN configurator**, and **code update manager**. The communication with a MN is done via TCP/IP (**2**). The CFEngine distributes management data within the WMN (**3**).

4.2 Mesh Node with WSN Manager

The **WSN manager** is located on every MN provides the management functionality for the different SSNs. It consists of three databases, the **MARWIS server** with three program modules and the CFEngine (as shown in Fig. 2(b)).

The **WSN information database** stores all information about the SNs and the WSN, such as topology (neighbours, address), and states of the SNs (battery, memory). The **program version database** stores all versions of all programs for all platforms, which can be installed on the SNs, and the **sensor value database** stores all data measured by the sensors. All databases are accessible by an API to get and store data (**1** in Fig. 2(b)).

CFEngine is responsible for distributing management data within the WMN (**2**). Communication within the WMN als well with the SSN is done over TCP/IP

(**3**). The **WSN monitor module** connects to the WSN information database and to the sensor value database in order to handle the requests from the management station. It also stores data coming from the SNs into the databases. The **WSN configurator module** is responsible for the configuration tasks. It queries properties from the SNs and stores them in the WSN information database. The **code update manager module** stores newly received program images (and related information) in the program version database and notifies the management station about available programs. Compression mechanisms or differential patches are used to reduce the amount of transmitted data. To execute the updating process, it transmits the image to the SN.

4.3 Sensor Node with SN Agent

As shown in Fig. 2(c), the management tasks are handled by a **SN agent**. It consists of a SN monitor, a SN configurator, and a code updater. The **SN monitor** handles the monitor requests by sending the values to the MN. The **SN configurator** executes the configuration requests and notifies the MN. The **code updater** is responsible for the code replacement on the SN. It receives the program image of the application or operating system and performs the update by loading the new module and replacing the old one. Finally, it informs the MN about the success of the update. Communication within the SNs is done over TCP/IP (**1**).

5 WSN Management Protocols

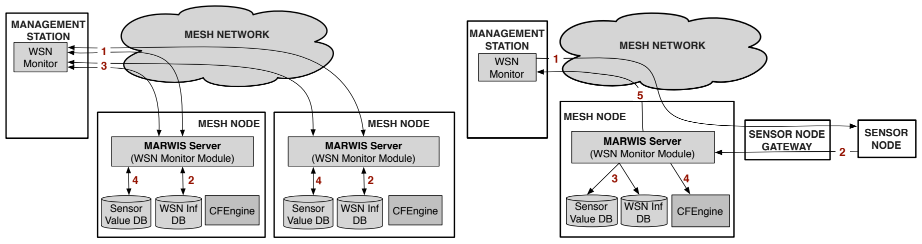
This section describes the management functionality in more detail. We consider the monitoring, configuring, and the code updating as important issues of our management architecture.

5.1 WSN Monitoring Protocol

Monitoring of the WSN can be performed in two ways. First, the management station explores the WMN and the subordinate SSNs. Alternatively, the user can query a selected sensor directly.

Fig. 3(a) shows how the management station queries the MNs about their SSNs **(1)**. The WSN monitor module queries the WSN information database **(2)**. Afterwards, the management station requests the current sensor values from every subordinate SN **(3)**, by querying the sensor value database **(4)**. All information from every SN is stored in the WSN information database and distributed in the whole WMN. Thus, we have a general view over the whole heterogeneous WSN. For displaying network topologies of SNs no additional transmissions to the SNs are required and the querying a MN is much faster than querying a SN.

Moreover, the user can request information from a SN directly and not query the WSN information database on the MN. This is shown in Fig. 3(b) and works as follows: the user requests information from a SN. The request is transmitted



(a) WSN monitor queries the MNs. (b) SN is queried directly.

Fig. 3. Monitoring the WSN

to the queried SN **(1)**, which sends the requested value back to the MN **(2)**. The WSN Monitor Module writes the new value into the database **(3)**, and distributes it within the WMN **(4)**. Finally, it sends the requested value to the WSN monitor **(5)**.

By using a WMN as backbone the number of hops is decreased (by dividing the WSN into SSNs). This means that the communication load of a direct request to a SN occurs mainly in the WMN. Thus, the request can be processed much faster and more energy-efficient. Overload and congestion in WSNs are prevented.

5.2 WSN Configuration Protocol

With the WSN configuration protocol the properties of the SNs as well as the network can be configured. Examples are switching sensors on/off, or changing routing tables. The procedure is similar to the WSN monitoring, but a configuration command is included in the request. As the SN configurator has a universal interface and hides the SN type specific characteristics, the packets with the configuration commands are independent of the node type. The heterogeneity of the WSN is hidden from the user, and therefore the configuration can be processed without knowledge of the specific node type. One or more sensor nodes can be targeted. When a new SN joins, first an initial network configuration is negotiated. Afterwards all available data is requested from the SN by the configuration module on the MN and propagated within the WMN.

5.3 Code Update Protocol

The code update protocol consists of three main subtasks. The new image of an application or the operating system is uploaded and stored in the program version database and distributed within the WMN. The management station is notified about the programs available. Finally, the image is transmitted to the SN performing the update. One or more sensor nodes can be targeted.

The main part of the protocol is the updating process of the SNs as shown in Fig. 4. The program version and the SNs are selected, sent to the involved MNs **(1)**, and checked by querying the WSN information database **(2)**. The

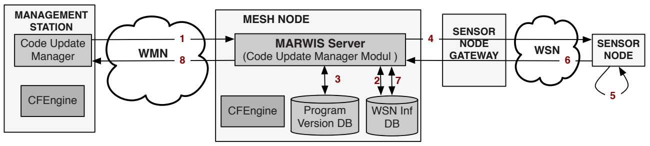


Fig. 4. The code update for the senor node is initiated

image is taken from the program version database **(3)**, and sent to the selected SN **(4)**. On the SN the update is performed **(5)** and acknowledged **(6)**. The WSN information database is updated **(7)** and finally the management station is notified **(8)**.

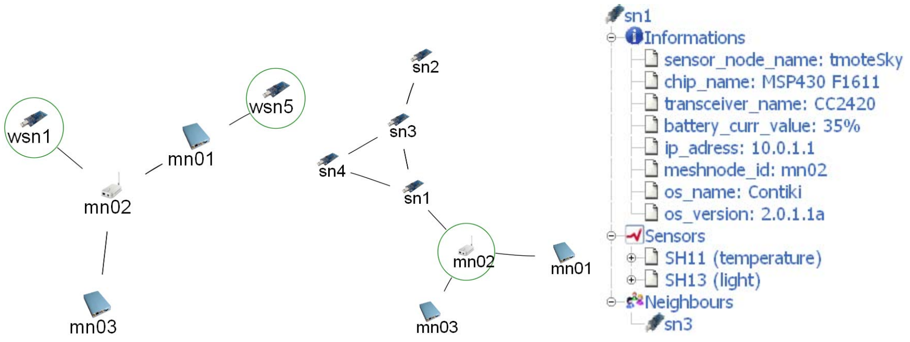
6 Implementation

This section describes the implementation of the first prototype of MARWIS. Four different components have to be realized: first, the management station with the **user interface** and the **management system for WMNs**; second, the MNs building a multi-hop WMN and providing the above described management functionality; third, the SN gateways to enable communication between the WMN and the WSN; and fourth, the different SNs running Contiki and building the heterogeneous WSN.

The management station consists of two computers. On one a Live-CD system is running, which contains all necessary programs and configurations to start the management software. The other simply provides a web browser for the user interface and connects over HTTPS to the computer running the Live-CD. This architecture is very flexible, because on one hand the Live-CD system requires only minimal hardware performance and can be booted on almost every standard computer. On the other hand, the system with the user interface can be located somewhere in the internet and has to support just a web browser.

The user interface visualizes the topology of the networks (Fig. 5) and the information about the SNs (Fig. 5(c)). By clicking on the hosting MN (e.g. mn01) the corresponding sensor subnetwork is shown (Fig. 5(b). By clicking on a SN (e.g sn01) the information about the SN, the installed sensors and the neighbors are shown (Fig. 5(c)). By selecting an element (e.g. the operating system version) it can be configured or an update can be initiated. The graphics are created by Graphviz 2.12 [26] using the neighborhood data from the WSN information database. Position data by GPS or distance estimation by radio signal analysis can be included in the neighborhood data for Graphviz to achieve an accurate topology illustration.

The Live-CD with the **management system for WMNs** contains a small Linux distribution (kernel 2.6.14.6) including all required applications, especially a HTTPS server for the connection with the user interface. The modules handling the management tasks and communication to the MNs are implemented as a



sn1
Informations
sensor_node_name: tmoteSky
chip_name: MSP430 F1611
transceiver_name: CC2420
battery_curr_value: 35%
ip_adress: 10.0.1.1
meshnode_id: mn02
os_name: Contiki
os_version: 2.0.1.1a
Sensors
SH11 (temperature)
SH13 (light)
Neighbours
sn3

(a) WMN. (b) WSN. (c) SN information.

Fig. 5. User interface

server written in C using sockets (MARWIS server). The databases are managed with sqlite3 [27]. The API for accessing the databases is implemented in C.

The WSN information database contains the following tables: meshnodes, meshnet, sensornodes, sensornet, sensors, swcomponents, and properties. The network tables list the neighbors of a MN responsible of a SN. The properties table is the central table, which contains the IDs of all possible properties, such as chip name, battery, memory, sensor values, software components. The tables sensornodes, sensors, and swcomponents contain the current values, and time stamps of the according properties. The sensor value database contain also the sensors table, and stores the whole history of the values. The program version database contains one table, which has meta information of a program's version (such as name, version, platform) and the link to the file. These databases are updated over our management data distribution system using CFEngine.

The CFEngine is designed to keep installations and configuration files in large computer networks up to date. In our approach, it distributes the topology and collected sensor data in the WMN. Therefore, a directory with files to share is specified in each MN. New files to distribute are copied into this directory. A neighbor node checks for new files and downloads them if necessary. Thus, the information is propagated through the whole network. Scripts can be executed handling the propagated data, e.g., write them into the according databases. MNs provide enough memory to store all this data (ca. 100MB for a 1000-node WSN running during one year with a measurement cycle of one hour).

On the MNs the same software as on the Live-CD system is running, except for the HTTPS server with PHP. The modules handling the management tasks and communication are also implemented in C. To communicate with the SSN a Serial Line Interface Protocol (SLIP) over the Linux TUN/TAP kernel module is used. SLIP connects the IP layer of the MN directly to the IP layer of the SN gateway. The SN gateway can communicate directly with the SSN.

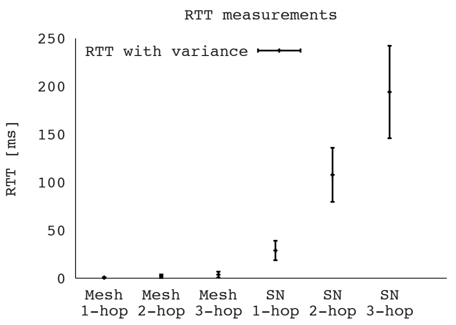
Contiki [24] is running on the SNs as a operating system. The code updater on the SN is responsible for the code replacement. Contiki works with loadable modules to allow replacing only parts of the operating system or applications. Except for the Contiki kernel, all modules can be replaced at run-time. The system has to be rebooted for kernel updates. The newly written applications have to fulfill just small constraints. To start and finish the application the functions _init() and _fini() are required and the application has to be complied as ELF (Executable and Linkable Format) loadable. The standard ELF or CELF (Compact ELF) is used. In contrast to ELF files CELF files are represented with 8 and 16-bit data types, which is adequate for 8-bit micro-controllers. Therefore, CELF files are usually half the size of the corresponding ELF file, but cannot be loaded by a standard ELF file handler. The code updater listens to the TCP port 6510 for new images of applications. On the MN side a small program is running, which sends a given image to port 6510 at a selected IP address.

After reception the referenced variables are checked and functions of the new application are linked and relocated by the Contiki dynamic Link Editor (CLE) and the application is copied to the ROM. Then the code updater starts the application using the _init() function.

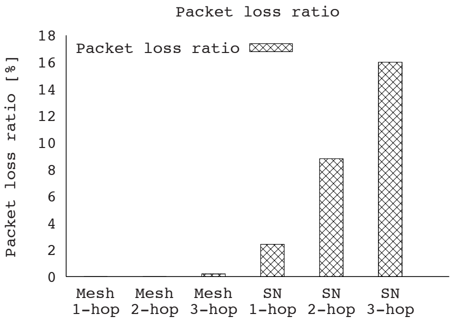
7 Evaluation

The further away a SN is located from the base station the higher the packet loss and the round trip time (RTT) are. Retransmissions for lost packets usually increase RTT and jitter. In our network architecture the WMN builds a fast backbone. Every SN in our network can connect with an average hop count of 2 to 3 hops to a mesh node. For investigation of the additional RTT and packet loss for increased hop count in WSN and WMN, we made experiments with ICMP echo request (ping). RTT time was evaluated with one, two and three hop sensor node and mesh node links. As MNs, an AMD Geode 233 CPU and 128 MB RAM with two 60 mW IEEE 802.11g interfaces, were used. The SN network tests were made with tmote sky nodes running a standard Contiki installation, as these SN platform features the fastest radio module (CC2420 with 1mW transmission power) and the highest amount of RAM. The SNs and MNs are located in different rooms. They are placed in such distance from each other distance between them is far from each other distanced that a node can only communicate with its one hop neighbors. Two hop neighbors are just recognized as interferences.

Fig. 6(a) shows the different RTT times over 500 measurements. An additional SN hop causes over 50 times longer RTT than a MN hop. Fig. 6(b) shows the packet loss over 500 packets, which is also much higher than on a mesh link. Although the measured values strongly depend on the hardware used as well as on the operating system and the communication protocols, the difference between a mesh and a sensor node hop is obvious. A large WSN with more than a dozen hops will be unserviceable. Moreover, it is possible to connect different SSNs with a WMN. With a MN backbone and directional antennas it is also possible to connect distant SN networks. Measurement where the sensor nodes are located within one room on one table are not realistic, less than the 1% of the packets get lost.



(a) RTT.



(b) Packet loss ratio.

Fig. 6. Evaluation

8 Conclusion

Most of the existing management approaches for WSNs are not dealing with heterogeneity. MARWIS, a management architecture for heterogeneous WSNs, solves this drawback. We showed that a complex heterogeneous WSN can be managed in an efficient way by using WMNs forming a backbone. We evaluated that the packet loss and the RTT is decreased significantly using a WMN as backbone, which is a precondition for an efficient management of heterogeneous WSNs. The MARWIS architecture with the modules and the used protocols provides the monitoring, the reconfiguration and the code updating of SNs in large heterogeneous WSNs.

References

1. Anwander, M., Wagenknecht, G., Staub, T., Braun, T.: Management of Heterogeneous Wireless Sensor Networks. 6. FG Sensornetzwerke, Aachen, Germany (July 2007)
2. Ruiz, L.B., Nogueira, J., Loureiro, A.: Manna: A Management Architecture for Wireless Sensor Networks. IEEE Communications 41(2), 116–125 (2003)
3. Marrón, P.J., Lachenmann, A., Minder, D., Hähner, J., Gauger, M., Saukh, O., Rothermel, K.: TinyCubus: A Flexible and Adaptive Framework for Sensor Networks. In: EWSN 2005, Istanbul, Turkey (January 2005)
4. Shen, C.C., Srisathapornphat, C., Jaikaeo, C.: An Adaptive Management Architecture for Ad Hoc Networks. IEEE Communication 41(2), 108–115 (2003)
5. Römer, K., Kasten, O., Mattern, F.: Middleware Challenges for Wireless Sensor Networks. SIGMOBILE Mob. Comput. Commun. Rev. 6(4), 59–61 (2002)
6. Heinzelman, W.B., Murphy, A.L., Carvalho, H.S., Perillo, M.A.: Middleware to Support Sensor Network Applications. IEEE Network 18(1), 6–14 (2004)
7. Liu, T., Martonosi, M.: Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In: PPoPP 2003, San Diego, CA, USA (June 2003)
8. Levis, P., Culler, D.: Maté: A Tiny Virtual Machine for Sensor Networks. In: ASPLOS-X 2002, San Jose, CA, USA (October 2002)

9. Aberer, K., Alonso, G., Kossmann, D.: Data Management for a Smart Earth. SIGMOD Rec 35(4), 40–45 (2006)
10. Han, C., Kumar, R., Shea, R., Srivastava, M.: Sensor Network Software Update Management: A Survey. Int. Journal of Network Man. 15(4), 283–294 (2005)
11. Brown, S., Sreenan, C.J.: Updating Software in Wireless Sensor Networks: A Survey. Technical Report UCC-CS-2006-13-07, University College Cork, Ireland (July 2006)
12. Wang, Q., Zhu, Y., Cheng, L.: Reprogramming Wireless Sensor Networks: Challenges and Approaches. IEEE Network 20(3), 48–55 (2006)
13. Reijers, N., Langendoen, K.: Efficient Code Distribution in Wireless Sensor Networks. In: WSNA 2003, San Diego, CA, USA (September 2003)
14. Jeong, J., Culler, D.: Incremental Network Programming for Wireless Sensors. In: SECON 2004, Santa Clara, CA (October 2004)
15. Koshy, J., Pandy, R.: Remote Incremental Linking for Energy-Efficient Reprogramming of Sensor Networks. In: EWSN 2005, Istanbul, Turkey (January 2005)
16. Marrón, P.J., Gauger, M., Lachemann, A., Minder, D., Saukh, O., Rothermel, K.: FlexCup: A Flexible and Efficient Code Update Mechanism for Sensor Networks. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, Springer, Heidelberg (2006)
17. Stathopoulos, T., Heidemann, J., Estrin, D.: A Remote Code Update Mechanism for Wireless Sensor Networks. Technical Report CENS-TR-30, University of California, Los Angeles, USA (November 2003)
18. Scatterweb: Platform for Self-configuring Wireless Sensor Networks. Last visit (March 2008), http://www.scatterweb.net
19. Tmote SKY: Reliable Low-power Wireless Sensor Networking Platform for Development. Last visit (March 2008), http://www.moteiv.com
20. BTnode: Flexible Platform for Fast-prototyping of Sensor and Ad-hoc Networks. Last visit (March 2008), http://www.btnode.ethz.ch
21. MICAz: The MICAz is a 2.4 GHz, IEEE/ZigBee 802.15.4, board used for low-power, wireless sensor networks. Last visit (March 2008), http://www.xbow.com
22. Staub, T., Balsiger, D., Lustenberger, M., Braun, T.: Secure Remote Management and Software Distribution for WMNs. In: ASWN 2007, Santander, Spain (May 2007)
23. Burgess, M.: A Tiny Overview of CFEngine: Convergent Maintenance Agent. In: MARS/ICINCO 2005, Barcelona, Spain (September 2005)
24. Dunkels, A., Grönvall, B., Voigt, T.: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In: EmNetS 2004, Tampa, FL, USA (November 2004)
25. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed Diffusion for Wireless Sensor Networking. ACM/IEEE Transactions on Networking 11(1), 2–16 (2002)
26. Graphviz: A Graph Visualization Software. Last visit (March 2008), http://www.graphviz.org
27. SQLite: A Self-contained, Serverless, Zero-configuration, Transactional SQL Database Engine. Last visit (March 2008), http://www.sqlite.org
28. Braun, T., Voigt, T., Dunkels, A.: TCP Support for Sensor Networks. In: WONS 2007, Obergurgl, Austria (January 2007)

Experiences from Two Sensor Network Deployments — Self-monitoring and Self-configuration Keys to Success

Niclas Finne, Joakim Eriksson, Adam Dunkels, and Thiemo Voigt

Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden
{nfi,joakime,adam,thiemo}@sics.se

Abstract. Despite sensor network protocols being self-configuring, sensor network deployments continue to fail. We report our experience from two recently deployed IP-based multi-hop sensor networks: one in-door surveillance network in a factory complex and a combined out-door and in-door surveillance network. Our experiences highlight that adaptive protocols alone are not sufficient, but that an approach to self-monitoring and self-configuration that covers more aspects than protocol adaptation is needed. Based on our experiences, we design and implement an architecture for self-monitoring of sensor nodes. We show that the self-monitoring architecture detects and prevents the problems with false alarms encountered in our deployments. The architecture also detects software bugs by monitoring actual and expected duty-cycle of key components of the sensor node. We show that the energy-monitoring architecture detects bugs that cause the radio chip to be active longer than expected.

1 Introduction

Surveillance is one of the most prominent application domains for wireless sensor networks. Wireless sensor networks enable rapidly deployed surveillance applications in urban terrain. While most wireless sensor network mechanisms are self-configuring and designed to operate in changing conditions [12,16], the characteristics of the deployment environment often cause additional and unexpected problems [8,9,11]. In particular, Langendoen et al. [8] point out the difficulties posed by, e.g., hardware not working as expected.

To contribute to the understanding of the problems encountered in real-world sensor network deployments, we report on our experience from recent deployments of two surveillance applications: one in-door surveillance application in a factory complex, and one combined out-door and in-door surveillance network. Both applications covered a large area and therefore required multi-hop networking.

Our experiences highlight that adaptive protocols alone are not sufficient, but that an approach to self-monitoring and self-configuration that covers more aspects than protocol adaptation is needed. An example where we have experienced the need for self-monitoring of sensor nodes is when the components used in low-cost sensor nodes behave differently on different nodes. In many of our experiments, radio transmissions triggered the motion detector on a subset of our nodes while other nodes did not experience this problem.

Motivated by the observation that self-configuration and adaptation is not sufficient to circumvent unexpected hardware and software problems, we design and implement a self-monitoring architecture for detecting hardware and software problems. Our architecture consists of pairs of probes and activators where the activators start up an activity that is suspected to trigger problems and the probes measure if sensor components react to the activator's activity. Callback functions enable a node to self-configure its handling of a detected problem. We experimentally demonstrate that our approach solves the observed problem of packet transmissions triggering the motion detector.

To find software problems, we integrate Contiki's software-based on-line energy estimator [5] into the self-monitoring architecture. This allows us to detect problems such as the CPU not going into the correct low power mode, a problem previously encountered by Langendoen et al. [8]. With two examples we demonstrate the effectiveness of the self-monitoring architecture. Based on our deployment experiences, we believe this tool to be very valuable for both application developers and system developers.

The rest of the paper is structured as follows. The setup and measurements for the two deployments are described in Section 2. In Section 3 we present our experiences from the deployments, including unexpected behavior. Section 4 describes our architecture for self-monitoring while the following section evaluates it. Finally, we describe related work in Section 6 and our conclusions in Section 7.

2 Deployments

We have deployed two sensor network surveillance applications in two different environments. The first network was deployed indoors in a large factory complex setting with concrete floors and walls, and the second in a combined outdoor and indoor setting in an urban environment.

In both experiments, we used ESB sensor nodes [14] consisting of a MSP430 microprocessor with 2kB RAM, 60kB flash, a TR1001 868 MHz radio and several sensors. During the deployments, we used the ESB's motion detector (PIR) and vibration sensor.

We implemented the applications on top of the Contiki operating system [4] that features the uIP stack, the smallest RFC-compliant TCP/IP stack [3]. All communication uses UDP broadcast and header compression that reduces the UDP/IP header down to only six bytes: the full source IP address and UDP port, as well as a flag field that indicates whether or not the header is compressed.

We used three different types of messages: *Measurement messages* to send sensor data to the sink, *Path messages* to report forwarding paths to the sink, and *Alarm messages* that send alarms about detected activity.

We used two different protocols during the deployment. In the first experiment, we used a single-hop protocol where all nodes broadcast messages to the sink. In the second experiment, we used a multi-hop protocol where each node calculates the number of hops to the sink and transmits messages with a limit on hops to the sink. A node only forwards messages for nodes it has accepted to be relay node for. A message can take several paths to the sink and arrive multiple times. During the first deployment only a few nodes were configured to forward messages, but in the second deployment any node could configure itself to act as relay node.

After a sensor has triggered an alarm, an alarm message is sent towards the sink. Alarm messages are retransmitted up to three times unless the node hears an explicit acknowledgment message or overhears that another node forwards the message further. Only the latest alarm from each node is forwarded.

2.1 First Deployment: Factory Complex

The first deployment of the surveillance sensor network was performed in a factory complex. The main building was about 250 meters times 25 meters in size and three floors high. Both floors and most walls were made of concrete but there were sections with office-like rooms that were separated by wooden walls. Between the bottom floor and first floor there was a smaller half-height floor. The largest distance between the sink and the most distant nodes was slightly less than 100 meters.

The sensor network we deployed consisted of 25 ESB nodes running a surveillance application. All nodes were either forwarding messages to the sink or monitored their environment using the PIR sensor and the vibration detector. We made several experiments ranging from a single hop network for measuring communication quality to a multi-hop surveillance network.

Single-Hop Network Experiment. We made the first experiment to understand the limitations of communication range and quality in the building. All nodes communicated directly with the sink and sent measurement packets at regular intervals.

Table 1. Communication related measurements for the first experiment

Node	Distance (meter)	Walls	Received	Sent (expected)	Sent (actual)	Reception ratio (percent)	Signal strength (avg,max)	
2	65	1 C	92	621	639	15%	1829	2104
3	21	1 W	329	587	588	56%	1940	2314
4	55	1 C	72	501	517	14%	1774	1979
5	33	2 W	114	611	613	19%	1758	1969
6	18	1 W	212	580	590	37%	1866	2230
7	26	2 W	347	587	588	59%	2102	2568
8	15	1 W	419	584	585	71%	2131	2643
9	25	1 W	194	575	599	34%	1868	2218
10	23	2 W	219	597	599	37%	1815	2106
11	17	1 W	331	591	593	56%	2102	2582
50	27	2 W	230	587	594	39%	1945	2334

Table 1 shows the results of the measurements. The columns from left are node id, distance from the sink in meters, number of concrete and wooden walls between node and the sink, number of messages received at the sink from the node, number of messages sent by the node (calculated on sequence number), actual number of messages sent (read from a log stored in each node), percentage of successfully delivered messages, and signal strength measured at the sink. Table 1 shows that as expected the ratio of received messages decreases with increasing distance from the sink. As full sensor coverage of the factory was not possible using a single-hop network, we performed the other experiments with multi-hop networks.

Multi-Hop Network Experiments. After performing some experiments to understand the performance of a multi-hop sensor network with respect to communication and surveillance coverage, we performed the final experiment in the first deployment during a MOUT (Military Operation on Urban Terrain) exercise with 15-20 soldiers moving up and down the stairs and running in the office complex at the top level of the building.

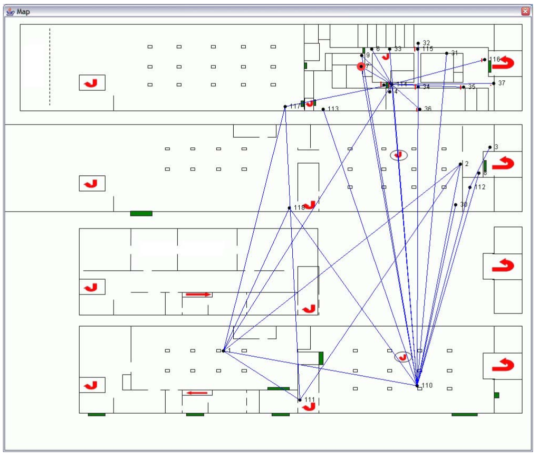


Fig. 1. Screenshot from the final experiment in the factory deployment illustrating placement of the sensor nodes during the surveillance and the paths used to transfer messages. All levels of the factory are shown with the top level at the top of the screenshot and the basement at the bottom. The office complex is on the right side at the top level in the figure.

Node 110 (see Figure 1) was the most heavily loaded forwarding node in the network. It had a direct connection to the sink and forwarded 10270 messages from other nodes during the three hour long experiment. During the experiment the sink received 604 alarms generated by node 110. 27 percent of these alarms were received several times due to retransmissions. Node 8 had seven paths to the sink, one direct connection with the sink, and six paths via different forwarding nodes. The sink received 1270 unique alarms from node 8 and 957 duplicates. Most of the other nodes' messages multi-hopped over a few alternative paths to the sink, with similar or smaller delays than those from node 110 and 8. This indicates that the network was reliable and that most of the alarms got to the server; in many cases via several paths. With the 25 sensor nodes we achieved coverage of the most important passages of the factory complex, namely doors, stairs, and corridors.

2.2 Second Deployment: Combined In-Door and Out-Door Urban Terrain

The second deployment was made in an artificial town built for MOUT exercises. It consisted of a main street and a crossing with several wooden buildings on both sides of the streets. At the end of the main street there were some concrete buildings. The distance between the sink and the nodes at the edge of the network was about 200 meters, making the network more than twice as long as in the first deployment.

The surveillance system was improved in two important ways. First, the network was more self-configuring in that there was no need for manually configuring which role each node should have (relay node or sensor node). Each node configured itself for relaying if the connectivity to sink was above a threshold. Second, alarm messages also included path information so that information of the current configuration of the network was constantly updated as messages arrived to the sink. Even with the added path information in the alarm messages, the response times for alarms in the network were similar to the response times in the first deployment despite that the distant nodes were three or four hops away from the sink rather than two or three. Using 25 nodes we achieved fairly good sensor coverage of the most important areas.

3 Deployment Experiences

When we deployed the sensor network application we did not know what to expect in terms of deployment speed, communication quality, applicability of sensors, etc. Both deployments were made in locations that were new to us. This section reports on the various experiences we made during the deployments.

Network Configuration. During the first deployment the configuration needed to make a node act as a relay node was done manually. This made it very important to plan the network carefully and make measurements on connectivity at different locations in order to get an adequate number of forwarding nodes. This was one of the largest problems with the first deployment. During the second deployment the network's self-configuration capabilities made deployment a faster and easier task.

The importance of self-configuration of the network routing turned out to be higher than we expected since we suddenly needed to move together with the sink to a safer location during the second deployment, where we did not risk being fired at. This happened while the network was deployed and active.

Unforeseen Hardware Problems. During radio transmissions a few of the sensor nodes triggered sensor readings which cause unwanted false alarms. Since we detected and understood this during the first deployment, we rewrote the application to turn off sensing on the nodes that had this behavior. Our long term solution is described in the next section.

Parameter Configuration. In the implementation of the communication protocols and surveillance application there are a number of parameters with static values set during early testing with small networks. Many of these parameters need to be optimized for better application performance. Due to differences in the environment, this optimization can only partly be done before deployment. Examples of such parameters are retransmission timers, alarm triggering delays, radio transmission power level, and time before refreshing a communication link.

Ground Truth. It is important for understanding the performance of a sensor network deployment to compare the sensed data to ground truth. In our deployments, we did not have an explicit installation of a parallel monitoring system to obtain ground truth, but in both deployments we received a limited amount of parallel feedback.

During the first deployment, the sensor networks alerted us of movements in various parts of the factory but since we did not have any information about the soldiers' current locations it was difficult to estimate the time between detection by the sensor nodes and the alarm at the sink. Sometimes the soldiers threw grenades powerful enough to trigger the vibration sensors on the nodes. This way, we could estimate the time between the grenade explosions and the arrival of the vibration alarm at the sink. During the second deployment we received a real time feed from a wireless camera and used it to compare the soldiers' path with the alarms from the sensor network.

Radio Transmission. During the first deployment we placed forwarding nodes in places where we expected good radio signal strength (less walls, and floors). We expected the most used path to the sink via forwarding nodes in the stairwells. However, most messages took a path straight through two concrete floors via a node placed at the ground floor below the office rooms where the sensors were deployed.

Instant Feedback. One important feature of the application during deployment was that when started, a node visualized its connection. When it connected, the node beeped and flashed all its leds before being silent. Without this feature we would have been calling the person at the sink all the time just to see if the node had connected to the network. This way, we could also estimate a node's link quality. The longer time for the node to connect to the network, the worse it was connected. We usually moved nodes that required more than 5 - 10 seconds to connect to a position with better connectivity.

4 A Self-monitoring Architecture for Detecting Hardware and Software Problems

To ensure automatic detection of the nodes that have hardware problems, we design a self-monitoring architecture that probes potential hardware problems. Our experiences show that the nodes can be categorized into two types: those with a hardware problem and those without. The self-probing mechanism could therefore possibly be run at start-up, during deployment, or even prior to deployment.

4.1 Hardware Self-test

Detection of hardware problems is done using a self-test at node start-up. The goal of the self-test is to make it possible to detect if a node has any hardware problems.

The self-test architecture consists of pairs of probes and activators. The activators start up an activity that is suspected to trigger problems and the probes measure if sensors or hardware components react to the activator's activity. An example of a probe/activator pair is measuring PIR interrupts when sending radio traffic. The API for the callbacks from the self tester for probing for problems, executing activators and handling the results are shown in Figure 3.

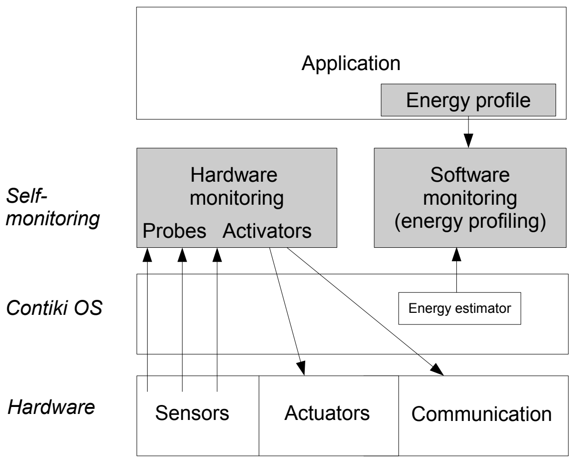


Fig. 2. Architecture for performing self-monitoring of both hardware and software. Self-tests of hardware are performed at startup or while running the sensor network application. Monitoring of the running software is done continuously using the built-in energy estimator in Contiki.

```
int probe();
void execute_activator();
void report(int activator, int probe, int percentage);
```

Fig. 3. The hardware self-test API

With a few defined probes and activators, it is possible to call a self-test function that will run all probe/activator pairs. If a probe returns anything else than zero, this is an indication that a sensor or hardware component has reacted to the activity caused by the activator. The code in Figure 4 shows the basic algorithm for the self-test. The code assumes that the activator takes the time it needs for triggering potential problems, and the probes just read the data from the activators. This causes the self-test to monopolize the CPU, so the application can only call it when there is time for a self-test.

The self-test mechanism can either be built-in into the OS or a part of the application code. For the experiments we implement a self-test component in Contiki on the ESB platform.

A component on a node can break during the network's execution (we experienced complete breakdown of a node due to severe physical damage). In such case the initial self-test will not automatically detect the failure. Most sensor network applications have moments of low action, and in these cases it is possible to re-run the tests or parts of the tests to ensure that no new hardware errors have occurred.

4.2 Software Self-monitoring

Monitoring the hardware for failure is taking care of some of the potential problem in a sensor network node. Some bugs in the software can also cause unexpected problems,

```
/* Do a self-test for each activator */
for(i = 0; i < activator_count; i++) {
  /* Clear the probes before running the activator */
  for(p = 0; p < probe_count; p++) {
    probe[p]->probe();
    probe_data[p] = 0;
  }
  for(t = 0; t < TEST_COUNT; t++) {
    /* run the activator and probe all the probes */
    activator[i]->execute_activator();
    for(p = 0; p < probe_count; p++)
      probe_data[p] += probe[p]->probe() ? 1 : 0;
  }
  /* send a report on the results for this activator-probe pair */
  for(p = 0; p < probe_count; p++)
    report(i, p, (100 * probe_data[p]) / TEST_COUNT);
}
```

Fig. 4. Basic self-test algorithm expressed in C-code

```
ENERGY_PROFILE(60 * CLOCK_SECOND,      /* Check profile every 60 seconds */
               energy_profile_warning, /* Call this function if mismatch */
               EP(CPU, 0, 20),         /* CPU 0%-20% duty cycle */
               EP(TRANSMIT, 0, 20),    /* Transmit 0%-20% duty cycle */
               EP(LISTEN, 0, 10));     /* Listen 0%-10% duty cycle */
```

Fig. 5. An energy profile for an application with a maximum CPU duty cycle of 20 percent and a listen duty cycle between 0 and 10 percent. The profile is checked every 60 seconds. Each time the system deviates from the profile, a call to the function *energy_profile_warning* is made.

such as the inability to put the CPU into low power mode [8]. This can be monitored using Contiki's energy estimator [5] combined with energy profiles described by the application developer.

4.3 Self-configuration

Based on the information collected from the hardware and software monitoring the application and the operating system can re-configure to adapt to problems. In the case of the surveillance application described above the application can turn off the PIR sensor during radio transmissions if a PIR hardware problem is detected.

5 Evaluation

We evaluate the self-monitoring architecture by performing controlled experiments with nodes that have hardware defects and nodes without defects. We also introduce artificial bugs into our software that demonstrate the effectiveness of our software self-monitoring approach.

5.1 Detection of Hardware Problems

For the evaluation of the hardware self-testing we use one probe measuring PIR interrupts, and activators for sending data over radio, sending over RS232, blinking leds and

```
/* A basic PIR sensor probe */
static int probe_pir(void) {
  static unsigned int lastpir;
  unsigned int value = lastpir;
  lastpir = (unsigned int) pir_sensor.value(0);
  return lastpir - value;
}

/* A basic activator for sending data over radio */
static void activator_send(void) {
  /* send packet */
  rimebuf_copyfrom(PACKET_DATA, sizeof(PACKET_DATA));
  abc_send(&abc);
}

/* Print out the report */
static void report(int activator, int probe, int trigged_percent) {
  printf("Activator %u Probe %u: %u%%\n", activator, probe, trigged_percent);
}
```

Fig. 6. A complete set of callback functions for a self test of radio triggered PIR sensor

beeping the beeper. The probes and activators are used to run the tests on ten ESB nodes of which two are having the hardware problems.

A complete but simplified set of probes, activators and report functions is shown in Figure 6. In this case, the results are only printed instead of used for deciding how the specific node should be configured.

As experienced in our two deployments, the PIR sensor triggers when transmitting data over the radio during the tests on a problem node. On other nodes the PIR sensors remain untriggered. Designing efficient activators and probes is important for the self-monitoring system. Figure 7 illustrates the variations in detection ratio when varying the number of transmitted packets and packet sizes during execution of the activator. Based on these results a good activator for the radio seems to be sending three 50 bytes packets.

5.2 Detection of Software and Configuration Problems

Some of the problems encountered during development and deployment of sensor network software are related to minor software bugs and misconfigurations that decrease the lifetime of the network [8]. Bugs such as missing to power down a sensor or the radio chip when going into sleep mode, or a missed frequency divisor and therefore higher sample rate in an interrupt driven A/D based sensor can decrease the network's expected lifetime. We explicitly create two software problems causing this type of behavior.

The first problem consists of failing to turn off the radio in some situations. Figure 8 shows the duty cycle of the radio for an application that periodically sends data. XMAC [1] is used as MAC protocol to save energy. XMAC periodically turns on the

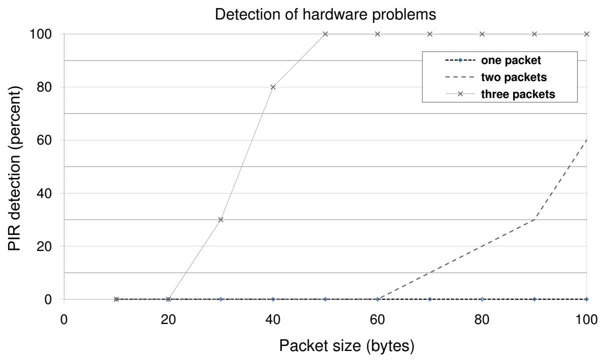


Fig. 7. Results of varying the activator for sending radio packets on a problem node. Sending only one packet does not trigger the PIR probe, while sending three packets with a packet size larger than 50 bytes always triggers the problem. On a good node no PIR triggerings occur.

radio to listen for transmissions and when sending it sends several packet preambles to wake up listeners. Using the profile from Figure 5 a warning is issued when the radio listen duty cycle drastically increases due to the software problem being triggered.

In the second problem the sound sensor is misconfigured causing the A/D converter to run at twice the desired sample rate. The node then consumes almost 50% more CPU time than normal. This misbehavior is also detected by the software self-monitoring component.

6 Related Work

During recent years several wireless sensor networks have been deployed the most prominent being probably the one on Great Duck Island [9]. Other efforts include glacier [11] and water quality monitoring [2]. For an overview on wireless sensor network deployments, see Römer and Mattern [13].

Despite efforts to increase adaptiveness and self-configuration in wireless sensor networks [10,12,16], sensor network deployments still encounter severe problems: Werner-Allen et al. have deployed a sensor network to monitor a volcano in South America [15]. They encountered several bugs in TinyOS after the deployment. For example, one bug caused a three day outage of the entire network, other bugs made nodes lose time synchronization. We have already mentioned the project by Langendoen that encountered severe problems [8]. Further examples include a surveillance application called “A line in the sand” [6] where some nodes would detect false events exhausting their batteries early and Lakshman et al.’s network that included nodes with a hardware problem that caused highly spatially correlated failures [7]. Our work has revealed additional insights such as a subset of nodes having a specific hardware problem where packet transmissions triggered the motion detector.

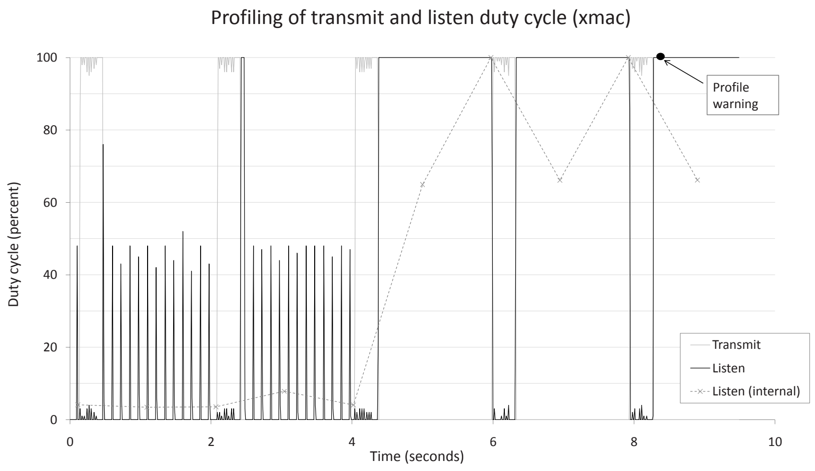


Fig. 8. Duty-cycle for listen and transmit. The left part shows the application's normal behavior with a low duty cycle on both listen and transmit. The right part shows the behavior after triggering a bug that causes the radio chip to remain active. The dashed line shows the listen duty cycle estimated using the same mechanism as the energy profiler but sampled more often. The next time the energy profile is checked the deviation is detected and a warning issued.

7 Conclusions

In this paper we have reported results and experiences from two sensor network surveillance deployments. Based on our experiences we have designed, implemented and evaluated an architecture for detecting both hardware and software problems. The evaluation demonstrates that our architecture detects and handles hardware problems we experienced and software problems experienced during deployments of other researchers.

Acknowledgments

This work was funded by the Swedish Defence Materiel Administration and the Swedish Agency for Innovation Systems, VINNOVA.

References

1. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In: ACM SenSys (November 2006)
2. Dinh, T.L., Hu, W., Sikka, P., Corke, P., Overs, L., Brosnan, S.: Design and Deployment of a Remote Robust Sensor Network: Experiences from an Outdoor Water Quality Monitoring Network. IEEE Congf. on Local Computer Networks, 799–806 (2007)
3. Dunkels, A.: Full TCP/IP for 8-bit architectures. In: Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS 2003), San Francisco, California (May 2003)

4. Dunkels, A., Grönvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of the First IEEE Workshop on Embedded Networked Sensors (IEEE Emnets 2004), Tampa, Florida, USA (November 2004)
5. Dunkels, A., Österlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: EmNets 2007: Proceedings of the 4th workshop on Embedded networked sensors, pp. 28–32 (2007)
6. Arora, A., et al.: A line in the sand: a wireless sensor network for target detection, classification, and tracking. Computer Networks 46(5), 605–634 (2004)
7. Krishnamurthy, L., Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Kushalnagar, N., Nachman, L., Yarvis, M.: Design and deployment of industrial sensor networks: experiences from a semiconductor plant in the north sea. In: ACM SenSys, pp. 64–75 (2005)
8. Langendoen, K.G., Baggio, A., Visser, O.W.: Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In: 14th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS) (April 2006)
9. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: First ACM Workshop on Wireless Sensor Networks and Applications (WSNA 2002), Atlanta, GA, USA (September 2002)
10. Marron, P.J., Lachenmann, A., Minder, D., Hahner, J., Sauter, R., Rothermel, K.: TinyCubus: a flexible and adaptive framework sensor networks. In: EWSN 2005, Istanbul, Turkey (2005)
11. Padhy, P., Martinez, K.K., Riddoch, A., Ong, H., Hart, J.: Glacial environment monitoring using sensor networks. In: Proc. of the Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden (June 2005)
12. Rhee, I., Warrier, A., Aia, M., Min, J.: Z-MAC: a hybrid MAC for wireless sensor networks. ACM SenSys, 90–101 (2005)
13. Römer, K., Mattern, F.: The design space of wireless sensor networks. IEEE Wireless Communications 11(6), 54–61 (2004)
14. Schiller, J., Ritter, H., Liers, A., Voigt, T.: Scatterweb - low power nodes and energy aware routing. In: Proceedings of Hawaii International Conference on System Sciences, Hawaii, USA (January 2005)
15. Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., Welsh, M.: Fidelity and yield in a volcano monitoring sensor network. In: Symposium on Operating Systems Design and Implementation (OSDI), Seattle, USA (2006)
16. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. ACM SenSys, 14–27 (2003)

Energy-Efficient Multiple Routing Trees for Aggregate Query Evaluation in Sensor Networks

Yuzhen Liu and Weifa Liang

Department of Computer Science, The Australian National University
Canberra, ACT 0200, Australia
{yliu,wliang}@cs.anu.edu.au

Abstract. In this paper we consider the problem of finding multiple routing trees in sensor networks for the evaluation of a class of aggregate queries including AVG, MIN, MAX, and COUNT with an objective to maximizing the network lifetime. Due to the NP hardness of the problem, we instead devise a heuristic algorithm for it. Unlike the previous work that focused on finding a single routing tree for query evaluation, we introduce the concept of multiple routing trees, and use these trees to evaluate aggregate queries, provided that different routing trees are used at different stages of the network lifetime. To evaluate the performance of the proposed algorithm, we conduct extensive experiments by simulation. The experimental results show that the proposed algorithm outperforms existing algorithms based on a single routing tree. We also prove that the approximation ratio of a known approximation algorithm for the identical energy case is a constant, and provide tighter lower and upper bounds on the optimal network lifetime for the non-identical energy case.

1 Introduction

In wireless sensor networks, each sensor periodically measures the physical environment around it and generates a stream of data. The processor within the sensor processes the data and relays the processed result to the other sensors. A wireless sensor network thus can be treated as a *sensor database* [1]. Several sensor database systems like TinyDB [2] have been proposed. Sensor databases allow users to pose queries to a special node (the base station) which then disseminates them over the network. In response to each query, each node evaluates its data against the query and transmits the matched data towards the origin of the query. As the matched data is routed through a routing tree consisting of all the nodes in the network, each relay node in the tree may apply one or more database operators (typically aggregation operators) on the data it received and/or sensed. Such data gathering query processing is referred to as *in-network processing*, which has been shown to be fundamental to achieve energy-efficient communication in data-rich, large-scale, yet energy-constrained sensor databases. The main constraint on the sensors is that they are equipped with energy-limited batteries, which limits the network lifetime and impairs the quality of the network. Therefore, energy conservation in sensor networks is of paramount importance. Energy-efficient routing trees built for in-network processing play a central role in such data gathering application [2,3,4,5].

Data gathering in sensor networks aiming to find a routing tree such that the total energy consumption is minimized has been extensively studied in literature [6,7,8,9]. Heinzelman *et al* [6] initialized the study of the problem by proposing a clustering protocol `LEACH`. The nodes in `LEACH` are grouped into a number of clusters in a self-organized way, where a clusterhead serves as a local 'base station' to aggregate data from its members and send the aggregated result to the base station directly. Lindsey and Raghavendra [7] studied the problem by providing an improved protocol `PEGASIS`, in which all the nodes in the network form a chain. One of the nodes in the chain is chosen as the head, and the head is responsible to report the aggregated result to the base station. In both [6] and [7], the measurement of energy consumption is not addressed explicitly. With the assumption that the transmission energy consumption at a node is proportional to the distance between the node and its parent, Tan and Körpeoğlu [8] studied the data gathering problem by proposing another protocol `PEDAP`, based on the above two solutions. `PEDAP` assigns weights to the communication links in the network and finds a minimum spanning tree rooted at the base station in terms of the total transmission energy consumption. Kalpakis *et al* [9] considered a generic data gathering problem with an objective to maximizing the network lifetime, and proposed an integer program solution and a heuristic solution. It should be noticed that all the data gathering methods mentioned above are based on the assumption that the length of the message transmitted by a relay node in a routing tree is independent of the lengths of its children messages, i.e. each node transmits the same volume of data to its parent no matter how much data it received from its children. We refer to this type of data gathering (aggregate) query as the *message-length independent aggregate query*, which is also called *fully aggregate query* in [10]. Such aggregate queries in databases include the popular operations `AVG, MIN, MAX, COUNT`, etc.

Unlike the previous work in [6,7,8,9,10] that either a dedicated routing tree is built for each individual query or a single shared routing tree is built for all the queries, we aim to build a series of routing trees for query evaluation such that the network lifetime is maximized, where network lifetime is referred to the time of the first node failure in the network [11]. Consider a sensor network with base station a, illustrated by Fig. 1(a). We assume that each node is assigned 1,500 units of initial energy. We further assume that each node only transmits a unit-length of data and the energy consumption in both transmitting and receiving a unit-length of data is one unit of energy for each aggregate query. The minimum degree spanning tree of Fig. 1(a) is shown in Fig. 1(b). If the optimal routing tree in Fig. 1(b) is used to evaluate aggregate queries until some node in the network runs out of energy, then the (optimal) maximum network lifetime is $\frac{1500}{1*4+1} = 300$. However, if the entire network lifetime is partitioned into several stages, then the network lifetime can be further prolonged, provided that a different routing tree is employed at each different stage. For the example in Fig. 1(a), we assume the network lifetime consists of two stages. We use another routing tree shown in Fig. 1(c) for the rest of aggregate queries after the current routing tree in Fig. 1(b) has been used for the first 240 aggregate queries, then the network lifetime is $240 + \frac{1500-240*(3+1)}{1*5+1} = 330$, which is longer than that delivered by using just a single spanning tree during the entire network lifetime.

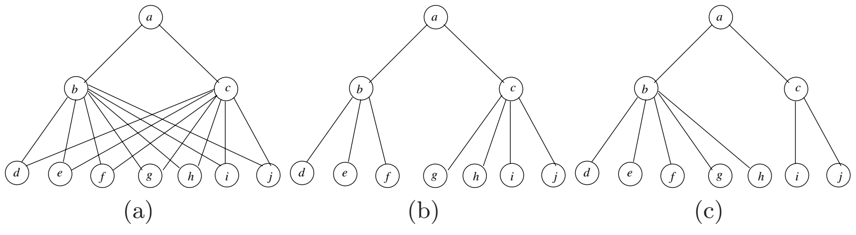


Fig. 1. (a)a sensor network (b)the optimal routing tree and (c)another routing tree

Motivated by the above example, we tradeoff the overhead of building routing trees and the difference of residual energies among the nodes, and introduce the concept of multiple routing trees. Instead of building a routing tree either for each query or for all queries, a routing tree just serves for a certain period of the network lifetime. Thus, a number of routing trees will be built for the evaluation of aggregate queries in sensor networks, and the energy overhead on building these trees can be reduced or alleviated, in comparison with that on building a dedicated routing tree for each individual query. On the other hand, we also balance the residual energies among the nodes by building another routing tree for the later queries, after a certain number of queries have been evaluated using the current routing tree.

2 Preliminaries

2.1 System Model

We consider a wireless senor network consisting of $n-1$ stationary homogeneous *sensor nodes* and a base station s distributed arbitrarily in a two dimensional region of interest. The network can be modelled by an undirected graph $G = (V, E)$, where V is the set of nodes with $|V| = n$ and there is an edge (u, v) in E if nodes u and v are within the transmission range of each other. The base station s is assumed to have unlimited energy supply. Each sensor node is equipped with an omni-directional antenna and can transmit messages with a fixed power level (its maximum power level). We assume that generating a unit-length of data consumes s_e amount of energy, transmitting a unit-length of data consumes t_e amount of energy, while receiving a unit-length of data takes r_e amount of energy. Clearly, $t_e \geq r_e$, and $s_e << \min\{t_e, r_e\}$. The transmission energy consumption of a sensor with transmission range R for transferring one unit-length data is R^2. Following the same assumption as in [10], there is time synchronization for evaluating aggregation queries. Time is divided into equal sized periods called *epochs*. At each epoch a leaf node transmits one unit of data to its parent. After a parent node has received the data from all its children, it aggregates the data and transmits the result to its parent in the next epoch.

2.2 Problem Definition

Given a wireless sensor network $G = (V, E)$, we assume that each sensor has a fixed transmission range and produces the sensed data as it monitors its vicinity periodically. Each sensor consumes the same amount of energy for one unit-length of data transmission, whereas its energy consumption of receiving data from its children is proportional to the number of its children. Specifically, we consider a class of message-length independent aggregate queries by finding multiple routing trees for them, with an objective to maximizing the network lifetime. For the sake of simplicity, we assume the message length transmitted by each node in the tree is one unit-length of data. The basic operation is to build one or multiple routing trees rooted at the base station and spanning all the other nodes. Each non-leaf node performs aggregation and forwards the partial aggregated result to its parent, the final query result will be available at the base station. It can be seen that the proposed algorithm can be easily to extended to the case where the message length transmitted by each node is an arbitrary l units of data with a minor modification, $l \geq 1$.

2.3 Algorithm for Finding a Single Routing Tree

We first briefly re-produce an approximation algorithm for finding such a routing tree that will be used for all aggregate queries during the entire network lifetime [10]. We then prove that the approximation ratio of the approximation algorithm for the identical energy case is a constant. We finally propose an approach to determine the lower bound and upper bound on the optimal network lifetime for the non-identical energy case.

Identical initial energy. If the initial energy at each node is identical, the network lifetime is determined by the maximum degree node in the routing tree, because the energy consumption of each node in the tree is determined by its transmission and reception energy consumptions, while the transmission energy consumption at each node is identical and the reception energy consumption of a node depends on how many children it has. Buragohain *et al* [10] concluded that finding a single optimal routing tree is equivalent to finding a Minimum Degree Spanning Tree (`MDST`) in the network, while the latter problem is to find a spanning tree such that the maximum node degree in the tree is minimized. The approximation ratio of the approximation algorithm given by Buragohain *et al* [10] is at least $\frac{1}{1+r_e}$, which depends on not only the transmission energy consumption ($t_e = 1$) but also the reception energy consumption r_e for a unit-length data. We now prove that the approximation ratio of their approximation algorithm is a constant by the following lemma.

Lemma 1. *Given a sensor network G in which each sensor has identical initial energy and transmission range, there is an approximation algorithm for the message-length independent aggregate query problem, and the network lifetime through the use of the routing tree delivered by the algorithm is at least 2/3 of the maximum (optimal) network lifetime.*

Proof. Let Δ^* be the maximum degree of the nodes in a minimum degree spanning tree in G and IE the initial energy at each sensor. Since $r_e \leq t_e$ and $\Delta^* \geq 2$, $\Delta^* + t_e/r_e \geq 3$. Following an algorithm due to Fürer and Raghavachari [12], referred to as `MDST`, there is an approximation solution for the minimum degree spanning tree problem, and the maximum node degree in the tree delivered by their algorithm is no more than $\Delta^* + 1$. Consequently, the maximum energy consumption of the maximum degree node in the routing tree is at most $r_e\Delta^* + t_e$ for each query, assuming one unit-length message will be transmitted, since the nodes in the tree have at most Δ^* children in the worst case. Let τ be the network lifetime delivered by the approximation algorithm `MDST` and τ_{opt} the maximum (optimal) network lifetime. We thus have

$$\frac{\tau}{\tau_{opt}} \geq \frac{\frac{IE}{r_e\Delta^*+t_e}}{\frac{IE}{r_e(\Delta^*-1)+t_e}} = \frac{r_e(\Delta^*-1)+t_e}{r_e\Delta^*+t_e} \geq 1 - \frac{1}{3} = 2/3.$$

Non-identical initial energy. It is reasonable to assume that the initial energy of each sensor is identical in the initial deployment of a sensor network. However, after a certain period of time, some sensors consume more energy than the others since they have been used as relay nodes to relay data for the others. As a result, the residual energies of different nodes will be different. This implies that the algorithm for the identical energy case is no longer applicable. For this case, Buragohain *et al* [10] proposed a novel reduction, which reduces the non-identical energy case to the identical energy case as follows.

Assume that the network lifetime τ is given in advance. Let $b(v)$ be the degree of node v in the tree. Then, the energy consumption at v for such an aggregate query is $r_e(b(v)-1)+t_e$ (assume a unit-length of data transferred by per node). Let $RE(v)$ be the residual energy at v at this moment. If the routing tree will be used for evaluating the rest of other aggregate queries, then, the maximum lifetime $\tau(v)$ of node v is bounded by $\tau(v) = \lfloor\frac{RE(v)}{r_e(b(v)-1)+t_e}\rfloor$. Obviously, $\tau \leq \min_{v\in V}\{\tau(v)\}$. Thus, the degree $b(v)$ of v in the routing tree is bounded by $b(v) = \lfloor\frac{RE(v)}{r_e\tau} - \frac{t_e}{r_e} + 1\rfloor$.

Given the sensor network $G(V, E)$ and the network lifetime τ with $|V| = n$, the algorithm in [10] first calculates $b(v)$ for each node $v \in V$. It then constructs an auxiliary graph $G' = (V \cup V_1, E \cup E_1)$ that is defined as follows. For each node $v_i \in V$, add $n - b(v_i)$ new nodes $v_{i_1}, v_{i_2}, \cdots, v_{i_{n-b(v_i)}}$ into V_1 and add an edge between v_i and v_{i_j} into E_1, $1 \leq j \leq n - b(v_i)$. Thus, the degree of each node $v \in V$ in G' is n, while the degree of every newly added node is one. An approximate, minimum degree spanning tree in G' is found afterwards. It finally prunes those nodes and edges incident to the nodes from the tree if the nodes are not in V. The resulting tree, referred to as *the degree-constrained spanning tree*, will be used as the routing tree. We refer to this algorithm as `NMDST`. However, in reality τ is unknown in advance. Buragohain *et al* [10] instead proposed an approach to find the optimal network lifetime by using the binary search on an interval of reasonable size. We here propose an approach to determine a narrow interval in which the optimal network lifetime τ_{opt} falls by the following lemma.

Lemma 2. *Let τ_{low} and τ_{upp} be the lower and upper bounds of the optimal network lifetime τ_{opt}. Then, $\tau_{low} = \min_{v \in V}\{\frac{RE(v)}{r_e(d(v)-1)+t_e}\}$ and $\tau_{upp} = \frac{\max_{v \in V}\{RE(v)\}}{r_e(\Delta^*-1)+t_e}$.*

Proof. Since the degree of any node in a spanning tree is no greater than its physical degree in the network, the lower bound then follows. We now deal with the upper bound on τ_{opt}. Let $b(v)$ be the degree of node v in a minimum degree spanning tree and $b(v_0) = \Delta^*$ where Δ^* is the maximum degree of nodes in the minimum degree spanning tree, $v_0 \in V$. Then

$$\tau_{opt} \le \min_{v \in V}\{\frac{RE(v)}{r_e(b(v)-1)+t_e}\} \le \frac{RE(v_0)}{r_e(\Delta^*-1)+t_e} \le \frac{max_{v \in V}\{RE(v)\}}{r_e(\Delta^*-1)+t_e}.$$

3 Heuristic Algorithm for Finding Multiple Routing Trees

In this section we propose a novel approach for the problem of concern to further prolong the network lifetime, if multiple rather than a single routing tree is employed during the different stages of the network lifetime. We start with two stages, and then consider K stages with a given $K(K \ge 2)$.

3.1 Finding Two Routing Trees

Assume that the initial energy at each node is identical, and the entire network lifetime consists of at most two stages. The proposed algorithm proceeds as follows.

In the first stage, an approximate, minimum degree spanning tree T_1 in G can be found using algorithm `MDST`, which will be used as the routing tree. After T_1 has been used for a certain period of time τ_1, a new spanning tree T_2 will be built, using algorithm `NMDST`, based on the current residual energy of each node. If the use of T_2 instead of T_1 will prolong the network lifetime further, the second stage proceeds, and T_2 will be used for evaluating the rest of aggregate queries. Otherwise, T_1 continues to be used until the end of network lifetime.

One fundamental issue related to this two-stage approach is to find the shifting time point τ_1, which is also the duration of the first stage. To find such a shifting time point, we check every τ' in the interval $[0, \tau]$ to see whether the inequality $\min_{v \in V}\{\frac{IE-[(d_{T_1}(v)-1)r_e+t_e]\tau'}{(d_{T_2}(v)-1)r_e+t_e}\} > \frac{IE}{(\Delta_{T_1}-1)r_e+t_e} - \tau'$ holds, where Δ_{T_1} is the maximum degree of nodes in T_1 and $d_{T_i}(v)$ is the node degree of v in T_i, $i = 1, 2$. If there is no such a τ', then, a single stage suffices. Otherwise, we select a τ'_0 that results in the longest network lifetime. In order to minimize the energy overhead on finding a shifting time point, we can use the binary search to find a shifting time point in the interval $[0, \tau]$. Thus, only $\log \tau$ routing trees are needed to be built.

3.2 Algorithm for Finding Multiple Routing Trees

In the following we propose an approach for finding $K \geq 2$ routing trees to prolong the network lifetime, which avoids the energy overhead on finding shifting time points of each stage as the above two-stage case. The idea is that the network lifetime is partitioned K stages. At each potential stage, the *quota of energy* assigned to each node is $1/K$ of the initial battery capacity. A routing tree in which each node is assigned at least the quota of energy will be used at each stage.

We now analyze the improvement of the network lifetime through the use of multiple routing trees in comparison to the use of a single routing tree. For simplicity, we assume that each query session only transfers a unit-length message, and the initial energy IE at each node is identical. Then, $\tau = \lfloor \frac{IE}{(\Delta-1)*r_e+t_e} \rfloor$, where Δ is the maximum degree of nodes in the approximate, minimum degree spanning tree. For convenience, in the rest of discussion we assume that the network lifetime is always an integral value and we ignore the floor of the computed value of the network lifetime. Clearly, the duration of the first stage, in which the approximate, minimum degree spanning tree T_1 in the network will be used, is $\tau_1 = \frac{\tau}{K}$, because the quota of energy assigned to each node v at this stage is $E_1(v) = IE/K$. After the first stage, a degree-constrained spanning tree T_i is constructed, using algorithm `NMDST` for each i of the remaining stages, assuming that the same quota of energy is assigned to each node at each stage, $i \geq 2$. If the duration of T_i is less than τ_1, T_1 will be used at stage i because the quota of energy of each node is at least IE/K and the duration of T_1 is at least τ_1. Otherwise, T_i should be used at stage i.

Let $b_i(v)$ be the degree of node v in the routing tree built at stage i and τ_i the duration of stage i. We have $\tau_i \geq \tau_1 \geq \frac{\tau}{K}$. Note that although the minimum quota of energy among the nodes at stage i is $\frac{IE}{K}$, the available energy at node v is actually $E_i(v) = \frac{IE}{K} + \Delta E_i(v)$, where $\Delta E_i(v)$ is the residual energy inherited from stage $(i-1)$, $2 \leq i \leq K$. Obviously, if $i = 2$, $\Delta E_2(v) = ((\Delta - b_1(v)) * r_e) * \tau_1$, which is the difference of energy consumption between the maximum degree node and node v in the routing tree T_1. Otherwise, $\Delta E_i(v) = E_{i-1}(v) - ((b_{i-1}(v)-1) * r_e + t_e) * (1/K + \delta_{i-1})\tau$, assuming that the duration of stage i is $\tau_i = (1/K + \delta_i)\tau$, $0 \leq \delta_i < 1$. Suppose that $\frac{E_i(v_{i_0})}{(b_i(v_{i_0})-1)*r_e+t_e} = \min_{v \in V}\{\frac{E_i(v)}{(b_i(v)-1)*r_e+t_e}\}$, we have $(1/K + \delta_i) * \tau = \frac{E_i(v_{i_0})}{(b_i(v_{i_0})-1)*r_e+t_e}$. Then,

$$\begin{aligned}
\delta_i &= \frac{1}{\tau}\left(\frac{E_i(v_{i_0})}{(b_i(v_{i_0})-1)*r_e+t_e}\right) - \frac{1}{K} \\
&= \frac{(\Delta-1)*r_e+t_e}{IE} * \frac{IE/K+\Delta E_i(v_{i_0})}{(b_i(v_{i_0})-1)*r_e+t_e} - \frac{1}{K} \\
&= \frac{(\Delta-1)*r_e+t_e}{(b_i(v_{i_0})-1)*r_e+t_e} * \left(\frac{1}{K} + \frac{\Delta E_i(v_{i_0})}{IE}\right) - \frac{1}{K} \\
&= \frac{1}{(b_i(v_{i_0})-1)*r_e+t_e} * \left(\frac{(\Delta-b_i(v_{i_0}))*r_e}{K} + \frac{((\Delta-1)*r_e+t_e)*\Delta E_i(v_{i_0})}{IE}\right) \\
&\geq \frac{1}{(d_G(v_{i_0})-1)*r_e+t_e} * \left(\frac{(\Delta-d_G(v_{i_0}))*r_e}{K} + \frac{((\Delta-1)*r_e+t_e)*\Delta E_i(v_{i_0})}{IE}\right) \qquad (1)
\end{aligned}$$

where $d_G(v)$ is the physical degree of node v in G. Thus, the value of δ_i depends on the network connectivity, the number of stages K, the initial battery capacity

IE, the transmission energy t_e and the reception energy r_e of each sensor. In particular, when the transmission range is reduced, the node degree $d_G(v)$ of v will decrease. This results in the reduction of the network connectivity and increase of the maximum degree Δ of nodes in the approximate, minimum degree spanning tree. Therefore, the gain of network lifetime at stage i is positive, and so is the entire network lifetime. Our later simulations in Section 4 indicates that the network lifetime delivered by using multiple routing trees is much longer than that by using a single routing tree when the transmission range is relatively smaller.

On the other hand, our objective is to maximize the entire network lifetime $\sum_{i=1}^{K} \tau_i$ rather than maximize the duration of each individual stage, where

$$\sum_{i=1}^{K} \tau_i = \tau_1 + \sum_{i=2}^{K} (\tfrac{1}{K} + \delta_i)\tau = (1 + \sum_{i=2}^{K} \delta_i)\tau. \tag{2}$$

Eq. (2) implies that the value of K should be as large as possible in order to maximize the network lifetime. However, there is a constraint on K from Inequality (1). It is obvious that δ_i is inversely proportional to the number of stages, and thus K is required to be as small as possible in order maximize the network lifetime at each stage. In addition, a larger K means that more frequent scheduling of using different routing trees is needed, which will incur an extra overhead on energy consumption. Therefore, there is a tradeoff between the choice of K and the prolonged network lifetime. The later experimental simulations confirm that K decreases with the growth of transmission range, since a longer transmission range implies a better network connectivity, and thereby reducing the maximum degree of the nodes in the approximate, minimum degree spanning tree and the difference of residual energy among the nodes become insignificant. The detailed algorithm for finding multiple routing trees is given below.

```
Algorithm. Multiple_Routing_Trees($G, IE, K$)
/* $G$ is the sensor network with initial battery capacity $IE$ at each node and */
/* $K$ is the number of potential stages */
begin
1. $\tau_1 \leftarrow$ MDST($G, IE/K$);
  /* $\tau_1$ is the network lifetime if an approximate, minimum degree spanning */
  /* tree is used at stage 1. */
2. for  $i \leftarrow 2$ to $K$ do
3.    $\tau_i \leftarrow 0$; /* $\tau_i$ is the duration of stage $i$ */
4.    for  each $v \in V$ do
5.       compute its residual energy $\Delta E_i(v)$;
         /* after the current tree has been used for $\tau_{i-1}$ units */
6.    endfor;
7.    $\tau_i^1 \leftarrow$ NMDST($G, IE/K + \Delta E_i()$);
      /* $\tau_i^1$ is the network lifetime delivered by a degree-constrained spanning */
      /* tree, using algorithm NMDST based on the quota of energy plus $\Delta E_i(v)$ */
      /* of each node $v$ at stage $i$ */
8.    if $\tau_1 < \tau_i^1$   then
      /* check whether the degree-constrained spanning tree will result in a */
      /* longer duration of stage $i$ */
```

```
                $\tau_i \leftarrow \tau_i^1$;
        else    $\tau_i \leftarrow \tau_1$;
    endif;
9. endfor;
end.
```

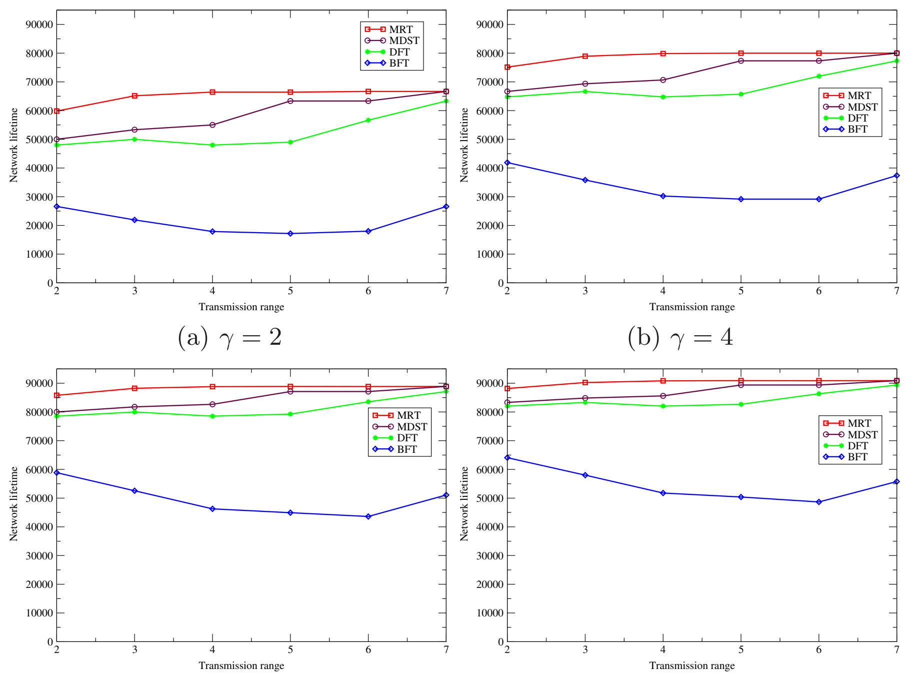
We refer to the above algorithm as algorithm `MRT` and have the following theorem.

Theorem 1. *Given a sensor network $G = (V, E)$ with identical initial battery capacity, assume that the transmission range of each sensor is identical. There is a heuristic algorithm for finding multiple routing trees for message-length independent aggregate queries. The network lifetime delivered by the proposed algorithm is longer than but at least as long as the one delivered by the algorithm for finding a single routing tree.*

4 Performance Evaluation

In this section we evaluate the performance of the proposed algorithm against existing algorithms through experimental simulations in terms of network lifetime. We assume that the monitored region is a 10×10 m^2 square in which 50 homogeneous sensor nodes are randomly deployed, by the NS-2 simulator. We also assume that the transmission range R of each sensor is from 2 to 7 with increment of 1. Two nodes can communicate with each other if and only if they are within the transmission range of each other, i.e., the Euclidean distance between them is no greater than R. We further assume that all the nodes have identical initial energy capacity $IE = 10^5$. Unless otherwise specified, we assume that the transmission energy consumption t_e for one unit-length of message is 1. Let $\gamma = t_e/r_e$, which is the *energy ratio* of the transmission energy consumption to the reception energy consumption for a unit-length data with $2 \leq \gamma \leq 10$. In the simulations, queries arrive one by one, and once a query arrives, it must be responded by the system, using the established routing tree. For simplicity, we assume that the answer to each query is a unit-length data as well. For each size of the network instance, the value shown in figures is the mean of 10 values obtained by running each algorithm on 10 randomly generated network topologies. In algorithm `MRT`, we limit the maximum number of various stages is $\lceil \sqrt{n} \rceil$, where n is the number of nodes in the sensor network and $n = 50$.

We first study the performance of the proposed algorithm `MRT` against the performance of the other three algorithms `MDST`, `BFT` and `DFT` by varying transmission ranges and energy ratios, where algorithms `MRT`, `MDST`, `BFT` and `DFT` are Multiple Routing Trees, Minimum Degree Spanning Tree, Breadth-First-Search Tree and Depth-First-Search Tree rooted at the base station respectively. As shown in Fig. 2(a), when the energy ratio $\gamma = 2$, algorithm `MRT` outperforms the others in terms of the network lifetime. For algorithm `MRT`, there is insignificant difference in the network lifetime when the transmission range R varies from 3 to 7, which implies that algorithm `MRT` can balance the energy consumption



(a) $\gamma = 2$ (b) $\gamma = 4$ (c) $\gamma = 8$ (d) $\gamma = 10$

Fig. 2. Comparison of network lifetime delivered by various algorithms

among the nodes in the network. Particularly, when $R = 7$, the network lifetime delivered by it is $\frac{10^5}{(0.5+1)} = 66666$, which is the optimal. This is because that the network connectivity is improved by increasing its sensor transmission ranges and the routing tree obtained is almost a chain. The network lifetime delivered by algorithm `MDST` increases with the growth of the transmission range R from 2 to 7, due to the fact that it tries to minimize the maximum degree of the nodes in the spanning tree, while the maximum degree node is the bottleneck of energy consumption among the nodes. With the improvement on network connectivity, the maximum degree of the nodes in the minimum degree spanning tree will reduce and thus the network lifetime is prolonged. Fig. 2(a) also shows that the difference of network lifetime delivered by algorithm `MRT` and `MDST` will diminish when the transmission range increases. The reason is that better connectivity improves the performance of algorithm `MDST` by reducing the maximum degree of nodes in the minimum degree spanning tree, and thus makes the network lifetime longer. When the transmission range R is 7, the maximum degree of nodes in minimum degree spanning tree becomes 2, and the network lifetime delivered by `MDST` reaches the maximum that is consistent with the network lifetime delivered by algorithm `MRT`. Meanwhile, it is observed that algorithm `BFT` is the worst among the algorithms, since the degrees of some nodes near to the tree root is maximized, while these nodes become the bottlenecks of energy consumption, thus they shorten the network lifetime. Similar to the performance of algorithm MDST, the degree of nodes of tree DFT drops with the improvement of network connectivity. Thus the network lifetime delivered by DFT increases with the growth of the transmission range.

When the energy ratio γ is 4, 8 or 10, the similar performance as the case where $\gamma = 2$ can be obtained, which is plotted in Fig. 2(b), (c) and (d). It can be seen that the performance of algorithm MRT is much better than that of algorithm MDST, especially for low connectivity sensor networks. The network lifetime delivered by algorithm MRT is almost the maximum one, which is $\frac{10^5}{(0.25+1)} = 80000$, $\frac{10^5}{(0.125+1)} = 88888$ or $\frac{10^5}{(0.1+1)} = 90909$ respectively when the transmission range R is no less than 3, whereas the network lifetime delivered by algorithm MDST is almost the optimal when the transmission range R is 7.

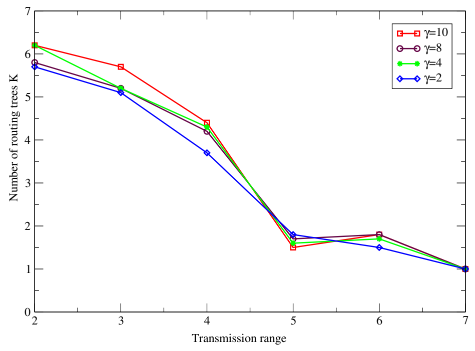


Fig. 3. The optimal number of routing trees in algorithm MRT

We then study the optimal number K of routing trees in algorithm MRT, and these trees will deliver the best possible network lifetime through experimental simulations. Recall that each value of K in Fig. 3 is the average of 10 different numbers of routing trees built for 10 different network topologies, given the transmission range R and the energy ratio γ. Fig. 3 shows that the optimal number of routing trees will decrease, when the transmission range R varies from 2 to 7 and the energy ratio γ is fixed. When the transmission range R is 2, the network lifetime is maximized if its network lifetime is partitioned upto six stages. This implies that more routing trees are needed for a low connectivity sensor network in order to maximize its network lifetime $\sum_{i=1}^{K} \tau_i$. When the transmission range is 7, the number of routing trees becomes 1, since the approximate, minimum degree spanning tree delivered by algorithm MDST now is a chain, when the nodes in the sensor network are highly connected with each other.

5 Conclusions

We have considered the evaluation of a class of message-length independent aggregate queries in sensor databases with an objective to maximizing the network lifetime. We first showed that the approximation ratio of a known approximation algorithm for the identical energy case is a constant, and provided the tighter lower and upper bounds on the optimal network lifetime for the non-identical energy case. We then introduced the concept of multiple routing trees to prolong the network lifetime further and devised a heuristic algorithm for finding such multiple routing trees. We finally conducted extensive experiments by simulation. The experimental results demonstrated that the performance of proposed algorithm significantly outperforms the existing ones that use only one single routing tree for the evaluation of such queries.

Acknowledgment. It is acknowledged that the work by the authors is fully funded by a research grant No:DP0449431 by Australian Research Council under its Discovery Schemes.

References

1. Govindan, R., Hellerstein, J.M., Hong, W., Madden, S., Franklin, M., Shenker, S.: The sensor network as a database. Technical Report 02-771, Computer Science Department, University of Southern California (September 2002)
2. Madden, S., Szewczyk, R., Franklin, M.J., Culler, D.: Supporting aggregate queries over ad hoc wireless sensor networks. In: Proc. 4th IEEE Workshop on Mobile Computing and System Applications, pp. 49–58. IEEE, Los Alamitos (2002)
3. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a tiny aggregation service for ad hoc sensor networks. ACM SIGOPS Operating Systems Review 36(SI), 131–146 (2002)
4. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proc. SIGMOD 2003, pp. 491–502. ACM Press, New York (2003)
5. Yao, Y., Gehrke, J.: The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Record 31(3), 9–18 (2002)
6. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proc. the Hawaii International Conference on System Sciences, pp. 3005–3014. IEEE, Los Alamitos (2000)
7. Lindsey, S., Raghavendra, C.S.: PEGASIS: Power-efficient gathering in sensor information systems. In: Proc. Aerospace Conference, pp. 1125–1130. IEEE, Los Alamitos (2002)
8. Tan, H.Ö., Körpeoğlu, İ.: Power efficient data gathering and aggregation in wireless sensor networks. ACM SIGMOD Record 32(4), 66–71 (2003)
9. Kalpakis, K., Dasgupta, K., Namjoshi, P.: Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. Computer Networks 42, 697–716 (2003)
10. Buragohain, C., Agrawal, D., Suri, S.: Power aware routing for sensor databases. In: Proc. INFOCOM 2005, pp. 1747–1757 (2005)
11. Chang, J.-H., Tassiulas, L.: Energy conserving routing in wireless ad hoc networks. In: Proc. INFOCOM 2000, pp. 22–31. IEEE, Los Alamitos (2000)
12. Fürer, M., Raghavachari, B.: Approximating the minimum-degree Steiner tree to within one optimal. J. Algorithms 17, 409–423 (1994)

A Collaborative Localization Scheme from Connectivity in Wireless Sensor Networks

Jichun Wang, Liusheng Huang, Xiang Li, He Huang, and Jianbo Li

Department of Computer Science and Technology
University of Science and Technology of China
{jichunw,christli,huang83,jbli}@mail.ustc.edu.cn, ls huang@ustc.edu.cn

Abstract. Automatic localization of sensor node is a fundamental problem in wireless sensor networks. For many applications, it is meaningless without relating the sensed data to a particular position. Many localization procedures have been proposed in the field recently. In this paper, we present a collaborative localization scheme from connectivity (CLFC) for wireless sensor networks. In this scheme, the connectivity information is used to improve the accuracy of position estimation. Relative positions between sensors are corrected to satisfy the constraints of connectivity. The scheme is composed by two phases: initial setup phase and collaborative refinement phase. In initial setup phase, DV-Hop is run once to get a coarse location estimation of each unlocalized sensor. In collaborative refinement phase, a refinement algorithm is run iteratively to improve the accuracy of position estimation. We compare our work via simulation with two classical localization schemes: DV-Hop and AFL. The results show the efficiency of our localization scheme. When compared with DV-Hop, estimation error of CLFC is reduced by 14% and 20% for random beacon deployment and fixed beacon deployment respectively. Furthermore, the proposed method CLFC is much better than the traditional mass-spring optimization based scheme AFL in terms of convergence rate. This results in significant saving in message complexity and computation complexity.

Keyword: Collaborative, Localization, Connectivity, Wireless sensor networks.

1 Introduction

Rapid advances in micro-electro-mechanical system are making the realization of large-scale wireless sensor networks (WSNs) a tangible task. WSNs can be deployed for various applications including environment monitoring, disaster relief, surveillance, and target tracking, so on and so forth. In many of these applications, sensed data is always meaningless without relating to its physical location. So it is very important to gain location of sensor node automatically.

For large-scale wireless sensor networks, the position of each sensor can not be predetermined without special localization equipments like GPS receiver. Furthermore, it is costly and not feasible to attach a special localization equipment on each sensor. For these and those reasons, sensors are always deployed in an ad-hoc manner and beacons are sparse due to deployment or cost constraints. In this case, most unlocalized sensor nodes can not contact with enough beacons directly. Although DV-Hop[9] solves the problem of localization for sparse beacon deployment sensor networks, the estimation result of DV-Hop is not accurate because proximities measured between sensors and beacons bias during multi-hop communication and relative positions between unlocalized sensors are not considered in localization scheme. Relative positions between sensors are considered in MDS[1,8] based localization schemes, but all of them are centralized or local centralized, in a large-scale wireless sensor networks, collecting global information is always very difficult. Sensors close to sink become energy exhausted quickly in centralized localization schemes because they have to support more forwarding traffic.

Considering problems mentioned above, we design an accurate and distributed localization system which uses connectivity information to correct the location of each sensor in wireless sensor networks. The scheme is composed by two phases: initial setup phase and collaborative refinement phase. In initial setup phase, DV-Hop is run once to get a coarse location estimation of each unlocalized sensor. In collaborative refinement phase, a refinement algorithm is run iteratively to improve the accuracy of position estimation based on connectivity information between unlocalized sensors.

Main contributions of this paper are the followings: First, we propose an accurate localization scheme utilizing connectivity information. In this scheme, locations of sensors are corrected to satisfy the relationships of connectivity between sensors. Second, considering the difficulty in collecting global information of large-scale wireless sensor networks, our scheme is fully distributed and does not require any additional infrastructure (ultra-sound, laser radiation, acoustic etc). Third, our method achieves faster convergence than traditional mass-spring optimization based localization scheme like AFL[2]. In our algorithm, potential energy of sensor nodes are used to evaluate the accuracy of position estimation. With the use of potential energy, the convergence rate of our scheme increases significantly, so the message complexity and computing complexity of our algorithm can be reduced.

The rest of this paper is organized as follows: the next section summarizes related works in localization for wireless sensor networks. Section 3 introduces our two-phase algorithm. Section 4 evaluates the performance of our approach through the comparison with previous works. Finally, we conclude in section 5.

2 Related Works

Many localization systems and algorithms have been proposed in the past few years. All these methods can be partitioned into two kinds: range-based solutions and range-free solutions. Range-based localization schemes use some kind of range or angle information to obtain location estimation. These measurement can be determined using one of four basic techniques: time of arrival[3] (TOA), time difference of arrival[4] (TDOA), angel of arrival[5] (AOA) and received signal strength indicator[6,7,13] (RSSI). With the knowledge of transmitter power and path loss model, the power of received signal can be used to determine the distance between sender and receiver. However, many factors affect range accuracy, such as multi-path reflection, non-line-of-sight condition and irregular signal propagation model. All these factors would make range estimation inaccurate. Although TOA, TDOA, AOA can achieve better accuracy, some additional infrastructures (ultra-sound, laser radiation, acoustic etc) are required in these systems.

Recently, several novel MDS-based localization schemes have been proposed in [1,8]. MDS is a data analysis technique, which can map n objects into an m dimensional embedding space ($m < n$ and usually, $m = 2, 3$). The representation of these objects in embedding space is called a relative map. The relative map can be transformed into an absolute map through scaling, rotation and reflection based on the coordinates of beacons. The drawback of these schemes is that they require availability of global information of all sensor nodes and all of them are centralized methods.

Niculescu et al.[9] proposed a distributed, hop by hop position algorithm, called DV-Hop. In DV-Hop, beacons flood messages to all sensors in networks. Every sensor maintains a minimum hop count to these beacons. For each sensor, average distance per hop can be obtained by exchanging message with beacons. Based on such information, distance between sensor and beacon can be derived. Then, with distances to at least three beacons, triangulation localization method can be used to determine absolute coordinate of each sensor. DV-Hop works well in sensor network with sparse beacon deployment, but the accuracy of DV-Hop is not good enough.

Savvides et al.[10] presented a collaborative multilateration localization method. In this method, unlocalized nodes estimate their own locations by using information of beacon locations that are several hops away and distances to neighboring nodes. To prevent the effect of error propagation and accumulation, the localization problem was formulated as a global non-linear optimization problem. Because the distance between unlocalized nodes and beacons biases as the number of hop between them increases, this method needs more beacons than other methods to work well.

In[2,11],localization problem is formulated as a system of spring. Each distance measurement is represented as a spring of different length jointing two nodes together. The expanded or compressed springs would have stored potential energy. Each node runs a relaxation procedure iteratively to minimize the potential energy of the system and then the optimal set of coordinates can be inferred. It is a range-based localization scheme and special hardware is required to measure distance between nodes.

3 Two-Phase Localization Method

In this section, we introduce the collaborative localization scheme. The localization scheme can be separated into two phases: initial setup phase and collaborative refinement phase. In the initial setup phase, DV-Hop is run once to get a coarse location estimation of each unlocalized sensor. In collaborative refinement phase, the refinement algorithm is run iteratively and cooperatively to improve the accuracy of location estimation of each sensor. The refinement algorithm only considers relative position information of sensors within k hops, where k is an adjustable parameter. Mass-spring optimization is introduced in the refinement phase to estimate the location of sensors using connectivity information within k hops. The connections between sensor nodes are equivalent to the spring displacement, the direction of resultant force and the potential energy are used to make iterative correction until the system converges to a stable state. Based on the estimation result of refinement algorithm and connections between sensor nodes, each sensor can calculate a constrained region of its own. The centroid of the constrained region is outputted as the final position estimation of sensor node.

3.1 Initial Setup-Phase

Considering a large-scale wireless sensor networks with sparse beacons deployment, most nodes in networks have no information about their own positions, with the exception of beacons. A very small percentage of unlocalized nodes can contact with beacons directly in networks. Furthermore, it is probable that none of unlocalized nodes in this networks can establish enough direct contact with beacons. So, we use DV-Hop in the initial setup phase to get a rough location estimation of sensor nodes.

Main steps of DV-Hop are as follows:

- At the first step of DV-Hop, each beacon floods its own position through the networks. So all nodes in the networks get the hop-count to all beacons.
- Once a beacon gets hop-count to all other beacons, it estimates the average length for one hop (called hop correction) and then broadcasts this information to its neighbors. When a sensor receives the first hop correction, it stores the information and broadcasts it to its neighbors. After that the sensor does not wait for subsequent hop correction and can start the next step of the algorithm.
- Finally, sensors perform a triangulation to three or more beacons to estimate their position according to the hop-count recorded for each known beacon and the hop correction. The resulting position estimate is likely to be coarse in terms of accuracy, but it provides an initial condition from which refinement algorithm can launch.

3.2 Collaborative Refinement-Phase

Given the initial position estimation of DV-Hop in the initial setup phase, mass-spring optimization is used in refinement phase to obtain more accurate position estimations by using the connectivity between sensor nodes. Since refinement must work in an ad-hoc manner, only the connectivity of local $k - hop$ (always,

$k = 2$ *or* 3) neighbors is considered. We demonstrate the estimation result comparison between $2-hop$ based localization and $3-hop$ based localization in Fig.1. Figure 1 shows that only the connectivity information of local $2-hop$ neighbors is acceptable. The refinement algorithm runs concurrently at each node and can be described as below.

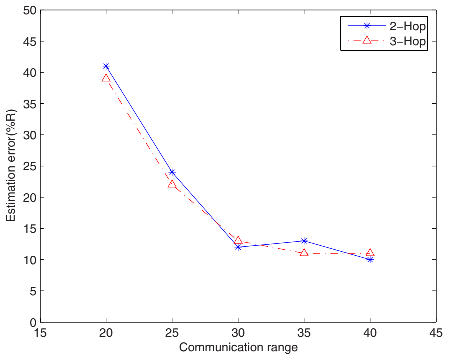


Fig. 1. Estimation error comparison between 2-hop based localization and 3-hop based localization

1. At the beginning, each node n_i broadcasts its position estimation p_i and receives position estimation from k-hop neighbors nearby.
2. For each node n_j within $k-hops$, let p_j represent the position estimation of node n_j, d_{ij} represent the Euclidean distance between p_i and p_j, $\overrightarrow{v_{ij}}$ be the unit vector in the direction from p_i to p_j, str_j represents the stretch factor of node n_j, the force $\overrightarrow{F_{ij}}$ is calculated as follows:
   If node n_j is one-hop neighbor of node n_i, the force is given by
$$\overrightarrow{F_{ij}} = \begin{cases} 0 & : \quad d_{ij} \leq R \\ \overrightarrow{v_{ij}} \cdot str_j \cdot (d_{ij} - R) & : \quad d_{ij} > R \end{cases} \tag{1}$$
   If node n_j is not one-hop neighbor of node n_i, the force is given by
$$\overrightarrow{F_{ij}} = \begin{cases} 0 & : \quad d_{ij} > R \\ \overrightarrow{v_{ij}} \cdot str_j \cdot (d_{ij} - R) & : \quad d_{ij} \leq R \end{cases} \tag{2}$$
   The resultant force on node n_i is given by
$$\overrightarrow{F_i} = \sum_{n_j \in N_{ik}} \overrightarrow{F_{ij}} \tag{3}$$
   Where, N_{ik} is the set of all $k-hop$ neighbors of node n_i and the total potential energy of node n_i is given by
$$E_i = \sum_{n_j \in N_{ik}} E_{ij} = \sum_{n_j \in N_{ik}} \|\overrightarrow{F_{ij}}\|^2 \tag{4}$$

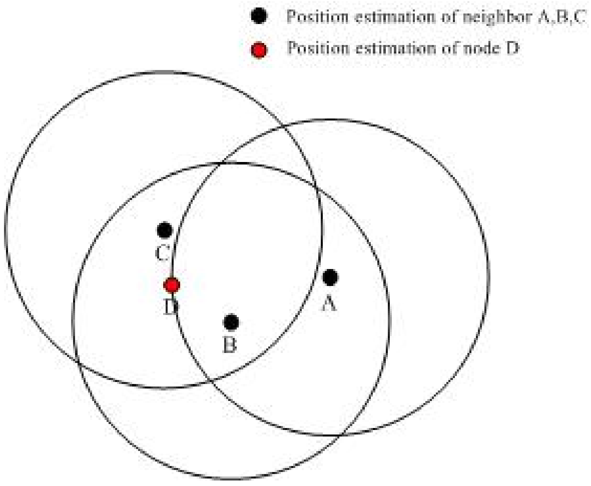
3. Based on resultant force $\overrightarrow{F_i}$ and potential energy E_i, the new position estimation of sensor node n_i can be determined by

$$p_i = p_i + \overrightarrow{v_i} \cdot \sqrt{\frac{E_i}{N}} \cdot \eta (0 < \eta < 1) \tag{5}$$

Where,$\overrightarrow{v_i}$ is the unit vector of $\overrightarrow{F_i}$ and N is given by

$$N = \sum_{n_j \in N_{ik}} str_j \tag{6}$$

Because beacon nodes are much more confident about their own positions, the stretch factor of beacon nodes should be larger than that of sensor nodes, in our algorithm,$str_j = 5$ if node n_j is a beacon and $str_j = 1$ if node n_j is not a beacon.



● Position estimation of neighbor A,B,C
● Position estimation of node D
C
D
A
B

Fig. 2. Position estimation converges near the border of constrained region

After a number of iterations mentioned above, most nodes' position estimations converge near the border of constrained region, just as shown in Fig.2. Because the average potential energy of node is almost zero when the position estimation approaches the border of constrained region. In order to locate node n_i at the centroid of constrained region, we randomly choose M samples within a circle centered at the current position estimation p_i and radius R. After choosing a sample s, its weight is determined using the neighborhood position estimation. The weight of a sample s for node n_i, $w_s(n_i)$ is computed as follows:

$$w_s(n_i) = \begin{cases} 1 & : \quad E_{\text{sample(s)}} \leq E_{p_i} \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7}$$

Finally, the position estimation of node n_i is given by

$$p_i(final) = \frac{\sum_{1 \le s \le M} w_s(n_i) \cdot p_{\text{sample(s)}}}{\sum_{1 \le s \le M} w_s(n_i)} \tag{8}$$

In order to avoid the denominator of equation above to be zero, we let $p_{\text{sample}(1)} = p_i$.

3.3 Analysis

In our simulation, sensor nodes are assumed to be deployed independently and randomly on a unit square plane with density ρ (that is, ρ is equal to $\frac{T}{A}$, where T is the total number of sensor nodes and A is the total surface area). The random deployment of sensor nodes can be modeled as a spatial homogeneous poisson point process. Let $Neighbor_s$ denote the set of neighbors heard by a sensor s. The probability that $|Neighbor_s| = k$ can be described as follows:

$$Pr(|Neighbor_s| = k) = \frac{(\rho\pi R^2)^k}{k!} e^{-\rho\pi R^2} \tag{9}$$

where R is the communication range of sensor node(we assume that each sensor node will transmit with the same radius R).

Based on network model described above, we can analyze the accuracy of localization result produced by our algorithm. The estimation error of our algorithm mainly comes from two source:

1. Distance estimation error between beacons and sensors is the main factor that influences the accuracy of the initial position estimates. According to Equation 9, we can derive that the expected connectivity of sensor nodes(called Con_s) to be $\rho\pi R^2$. Let d_{hop} denote the expected distance in one hop, Kleinrock and Silvester[12] showed that d_{hop} depends only on the expected degree of connectivity, not the total number of nodes.

$$d_{hop} = \sqrt{\frac{Con_s}{\rho\pi}} \cdot (1 + e^{-Con_s} - \int_{-1}^{1} e^{\frac{-Con_s}{\pi}(arccos t - t\sqrt{1-t^2})} dt) \tag{10}$$

   As the expected connectivity increases, the probability of nodes along the straight-line path increase rapidly. So when the density ρ increases, distance estimation error between beacons and sensors can be reduced. So the accuracy of our algorithm increases when the degree of connectivity increases.
2. Errors propagate fast through the whole network during the running of refinement algorithm. If the diameter of network is d, then an error introduced by a node in step p has affected every node in the network by step p+d. Just as shown in Equation 5, we use average potential energy to evaluate the quality of a location estimate. Lower average potential energy values indicate that more connectivity constrains have been satisfied. Intuitively, a good location estimate only needs minor adjustment, so position updates of sensor nodes with low average potential energy become slow in our refinement algorithm.

4 Performance Evaluations

4.1 Simulation Model

We implement our proposed localization scheme as C++ code running under the control of the OMNeT++. In order to exam the performance of our position method, different beacon deployment strategies are considered in our simulation. The first strategy is that four beacons are placed at the vertex of the square (fixed beacon deployment), while the second strategy is that four beacons are randomly placed in the square region (random beacon deployment). In our simulations, we use the following parameters to measure the accuracy of our location algorithm:

– Communication Range: the communication range of beacons and sensors. (we assume all nodes are homogenous, they have the same communication range R).
– Connectivity: average number of neighbors of unlocalized sensors.
– Distance Error: distance error is the Euclidian distance between the estimation location and real location of sensor.
– Estimation Error: estimation error is normalized to the communication range in order to compare simulation result conveniently.
– Random beacon deployment: four beacons are randomly placed in a 100-by-100 square region.
– Fixed beacon deployment: four beacons are fixed at the vertex of a 100-by-100 square region.

4.2 Estimation Error When Varying Communication Range

Figure 3 shows the effect of varying communication range on the performance of DV-Hop and CLFC. From Fig.3, we see that estimation error decreases when communication range increases. This can be explained by that when communication range increases, average number of nodes heard by unlocalized sensor increases. This implies that more constraints can be used to determine the location of unlocalized sensor, so the accuracy of estimation result improves.

4.3 Estimation Error When Varying Connectivity

Figure 4 explores the effect of connectivity on the performance of DV-Hop and CLFC. As seen in Fig.4, the accuracy of CLFC increases when the average connectivity increases, because more relative positions can be used in refinement algorithm. Position estimations of our method outperform DV-Hop significantly in scenarios with an average connectivity level of 20 or greater, estimation error decrease by 20% after the refinement.

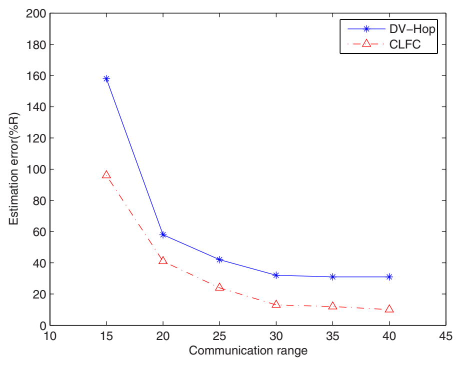


Fig. 3. Comparison of estimation error when varying communication range

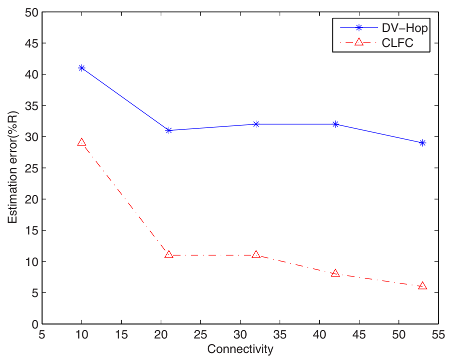


Fig. 4. Comparison of estimation error when varying connectivity

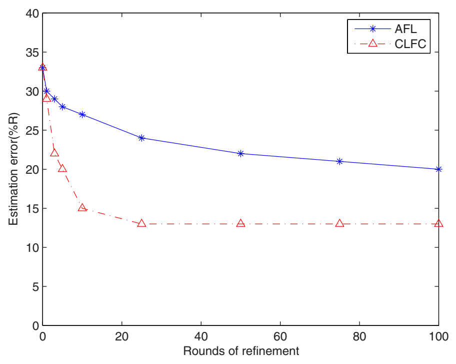


Fig. 5. Comparison of convergence between AFL and CLFC

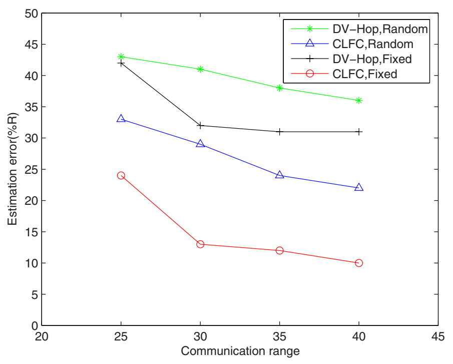


Fig. 6. Comparison of estimation error based on different beacon deployment

4.4 Estimation Error When Varying Rounds of Refinement

Figure 5 presents the comparison of convergence rate between AFL and CLFC. From Fig.5, we can see that CLFC converges much faster than AFL. CLFC needs only twenty rounds before the system converges to a stable state, while AFL needs more than one hundred rounds. So, our method can reduce message complexity and computing complexity significantly.

4.5 Estimation Error for Two Different Kinds of Beacon Deployment

In Fig.6, the effect of beacon deployment on estimation error is considered. When compared with random beacon deployment strategy, fixed beacon deployment strategy gets a gain of 10% decrease of estimation error. Because DV-Hop provides a more accurate estimation of unlocalized sensor when four beacons are placed at the vertex of the square region, the accuracy of initial position estimations influences the accuracy of iterative refinement algorithm.

5 Conclusions

In this paper, we propose and evaluate a collaborative localization scheme. In this scheme, relative positions between sensors and connectivity information are used to increase the accuracy of localization of sensors. Different from previous works based on MDS which are centralized or locally centralized, the scheme proposed by us is fully distributed. Compared with AFL, our method can gain faster convergence. We implement our algorithm on simulation. The result show that the proposed collaborative localization scheme can get much more accurate position estimations than DV-Hop and converges much faster than AFL.

Acknowledgements

We really appreciate the valuable advices given by our teammates and other researchers. And this paper is supported by the National Grand Fundamental Research 973 Program of China under Grant No. 2006CB303006, the National Grand Fundamental Research 973 Program of China under Grant No. 2007CB316505, the Knowledge Innovation Project of the Chinese Academy of Sciences, the Dean Fund of the Chinese Academy of Sciences, and the Doctoral Program of Higher Education of China under Grant No.20070358075.

References

1. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.P.J.: Localization from mere connectivity. In: Proc. Mobihoc 2003, June 2003, pp. 201–212 (2003)
2. Priyantha, N.B., Balakrishnan, H., Demaine, E., Teller, S.: Anchor-free distributed localization in sensor networks. Technical Report 892, MIT Laboratory for Computer Science (April 2003)
3. Wellenhoff, B.H., Lichtenegger, H., Collins, J.: Global Position System: Theory and Practice, 4th edn. Springer Verlag, Heidelberg (1997)
4. Savvides, A., Han, C.C., Srivastava, M.B.: Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In: Proceedings of Mobile Computing and Networks, Rome, Italy, July 2001, pp. 166–179 (2001)
5. Niculescu, D., Nath, B.: Ad Hoc Position System using AoA. In: Proceedings of the IEEE INFOCOM, San Francisco, pp. 1734–1743 (2003)
6. Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-Based User Location and Tracking System. In: Proceedings of IEEE INFOCOM, Tel-Aviv,Israel, vol. 2, pp. 775–784 (2000)
7. Lorincz, K., Welsh, M.: MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. In: Proceedings of International Workshop on Location and Contex-Awareness, Berlin, Germany, pp. 63–82 (2005)

8. Shang, Y., Ruml, W.: Improved MDS-based localization. In: IEEE Proc. Infocom 2004, March 2004, pp. 2640–2651 (2004)
9. Niculescu, D., Nath, B.: Ad hoc Positioning System (APS). In: Proceedings of IEEE Globecom, New York, USA, pp. 2926–2931 (2001)
10. Savvides, A., Park, H., Srivastava, M.: The Bits and Flops of the n-Hop Multilateration Primitive for Node Localization Problems. In: Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA 2002), September 2002, pp. 112–121 (2002)
11. Pandey, S., Prasad, P., Sinha, P., Agrawal, P.: Localization of Sensor Networks Considering Energy Accuracy Tradeoffs. IEEE CollaborateCom., 1–10 (December 2005)
12. Kleinrock, L., Silvester, J.: Optimum transmission radii for packet radio networks or why six is a magic number. In: Proc. of the IEEE National Telecommunications Conf. Birmingham, pp. 431–435. IEEE Press, Los Alamitos (1978)
13. Yedavalli, K., Krishnamachari, B.: Sequence-Based Localization in Wireless Sensor Networks. IEEE Transactions on mobile computing 7(1), 81–94 (2008)

Author Index