Estimations of Additional Delays for Mobile Application Data from Comparative Output-Input Throughput Analysis

Katarzyna Wac, *Member, IEEE*, Markus Fiedler, *Member, IEEE*, Richard Bults, Hermie Hermens

***Abstract*— Mobile devices with ever increasing functionality are an important driving force behind innovative mobile applications that enrich our daily life. The ubiquitous availability of wireless data communication networks is an additional driving force. Their ability to support application data flows is one of the performance criteria for successful mobile application deployment. Nevertheless, the quantitative impact of this performance is unknown and practically infeasible to determine at real-time at the application-level due to mobile device resource constraints. We research practical methods for measurement-based application-level performance evaluation of data communication networks that support mobile application data flows. In this paper, we apply the lightweight Comparative Output-Input Analysis (COIA) method that estimates *additional delay* at observation time scale of interest (e.g. 1 s) induced on the application data flow. The additional delay is the amount of delay exceeding non-avoidable, minimal end-to-end data delay caused by data communication network propagation and transmission delays. We propose five COIA methods to estimate the additional delay. We validate their accuracy with measurements obtained from an m-health application, namely health telemonitoring provided by the MobiHealth system. Despite their simplicity, our methods prove to be accurate in relation to the observation time scale of interest, and robust under a variety of network conditions. The methods offer novel insights into the impact of data communication network performance on the application data flow delay behaviour.**

***Index Terms*— delay estimation, interactive systems, mobile communication, quality assurance, performance management**

I. Introduction

EMERGING wireless network technologies and miniature personalized networked devices enable the provision of new mobile applications that support daily activities of their users [1]. These applications aspire to deliver services to their users 'anywhere-anytime-anyhow' while fulfilling their Quality of Service (QoS) and Quality of Experience (QoE) requirements [2], e.g. low application response times. However, the success of the service delivery depends heavily on the performance provided by the underlying network infrastructures [1–3]. As these services operate in heterogeneous networking environments, while the user is on the move, the knowledge of the overall network performance at a given user location and time is required by the service to optimize user's experience. Particularly, for interactive mobile applications, exchanging data between spatially dispersed mobile and fixed nodes, the end-to-end data delays and their hardly predictable, sudden increases, which we henceforth denote as *additional delays*, are critical performance measures [2, 3]. However, they are difficult to be measured at application run-time due to limited node capabilities and nodes' clock synchronization issues [4].

We propose five methods for run-time estimation of additional delays based on *Comparative Output-Input Analysis* (COIA). COIA builds upon the comparative analysis of application-level throughput at the sender and receiver nodes. We assume a 'black-box' view of the application on the underlying heterogeneous network infrastructure; the application can just observe the consequences of network (mis-)behavior, neither having a possibility to influence it, nor being able to exploit network-level measurements and/or feedback. We also assume an absence of precise clock synchronization at the nodes. We draw from the stochastic fluid flow model [5, 6] to analyze the application data traffic at small time scales, e.g. a second; the underlying time scale is dictated by data delivery deadlines.

We evaluate the accuracy of the proposed methods under a variety of network conditions, based on data traces from an existing interactive mobile application, a health telemonitoring/-treatment application provided by the MobiHealth system [7, 8]. Mobile healthcare applications pose strict application-level QoS requirements, since a patient in an emergency may require an immediate system response [9], e.g., activating his wearable defibrillator [10].

This paper is structured as follows. Section II provides concepts for modelling of additional delay, while Section III introduces the proposed estimation methods. Section IV provides accuracy evaluation results for these methods with data traces from the MobiHealth system. Section V concludes upon our research and draws future areas of work.

II. Modelling of the Additional Delay

A. The Concept of Additional Delay

The *additional delay* T^{add} is the amount of delay that exceeds the minimal possible end-to-end delay $T^{\min}$. While

Manuscript received September 13, 2009. This work is partially supported by the EU-COST Action "Data Traffic Monitoring and Analysis" (IC0703) and the Swedish Knowledge Foundation research project "Quality of Experience based Cross-Layer Design of Mobile Video Systems".

K. Wac is with Carnegie Mellon University, USA and Twente University, 7500 AE Enschede, NL (phone: +31-53-489-3743; fax: ext. 4524; e-mail: katewac@cs.cmu.edu).

M. Fieder is with Blekinge Institute of Technology, 37179 Karlskrona, Sweden (e-mail: markus.fiedler@bth.se).

R. Bults and H. Hermens are with University of Twente. R. Bults is also with MobiHealth BV, 7500 AE Enschede, NL (e-mail: {r.g.a.bults, h.hermens}@utwemte.nl).

978-1-4244-5367-2/10/$26.00 ©2010 IEEE

$T^{\min}$ is dominated by inherent, static propagation, transmission and processing delays, T^{add} obeys a dynamic random process dominated by transmission conditions as well as competition from other traffic and processes [11]. Assuming that data is sent by the sender application entity at time t^{in} and received by the receiver's entity at time t^{out}, T^{add} is obtained as

$$T^{\text{add}} = t^{\text{out}} - t^{\text{in}} - T^{\min}. \quad (1)$$

Ideally, the additional delay vanishes. However, as it grows, the more pronounced its negative effect on the end-to-end performance becomes, in particular if T^{add} varies a lot. Most interactive applications suffer from T^{add} exceeding certain thresholds; i.e. voice samples can be considered lost if they miss their delivery deadlines, and user perceived waiting times might grow beyond user patience levels [12]. Ideally, the additional delay of each and every application-level data message should be monitored in order to get a clear view of the disturbances that the data delivery process is exposed to. This is not feasible. The mobile node might not be able to trace and timestamp each packet in real-time due to scarce processing resources [3]. There are also limitations regarding the exactness of the timestamps, which might be too limited in resolution, incorrect due to processing times [13], and difficult to compare due to clock synchronisation issues [14].

A certain amount of variation of T^{add}, also denoted as jitter, is expected for packet-based data delivery. Each link and node leaves its footprint on the inter-packet-timing within a packet stream, which means that the timely behaviour of traffic at the receiver does not match the one at sender. For instance, round-trip delays over a non-disturbed UMTS network usually jitter by ± 10 ms [15]. These variations are not necessarily problematic from the application viewpoint. However, values of T^{add} in a range of one to several hundred milliseconds are perceptible [12], especially for interactive applications such as gaming [16]. These T^{add} values stem for example from congestion within access or core networks, or from temporarily bad radio-network conditions, implying the need to resend lost or corrupted data. From the end-user point of view, the latter looks like a sudden loss of capacity, followed by a "burst" of data arriving at the receiver with a much smaller spacing in time than they were sent [17, 18]. It is thus important to capture and handle additional delays exceeding the expected variation thresholds. In the next subsection, we discuss a model that is able to provide this functionality.

B. Application Flow Model

A model that has shown to be capable of discerning between less critical delays on a packet level and more critical delays on a burst level is the *fluid flow model* [5, 6]. So far it has been used for analysis of multiplexing in fast packet-switched networks and the consequences of temporary mismatches between capacity demand and availability, leading to considerable queuing on time scales beyond the packet scale. This model is based on the analysis of instantaneous workload instead of modelling the packet occurrence and length processes. In other words, the packets are "fluidified" in time and the intensity of the flow is quantified by the *data rate* $R(t)$, also denoted as *throughput* and measured in bits per second (bps), Bytes per second (Bps) or – in case packets of equal length are used – packets per second (pps). In general, $R(t)$ is a function of time and can be defined either on a continuous time scale as a derivative of the *cumulative workload* $W(t)$ passing a point of reference by time,

$$R(t) = \frac{\mathrm{d}}{\mathrm{d}t} W(t), \quad (2)$$

or on a discrete time scale as the workload observed at a point of reference during averaging interval i of duration ΔT, divided by ΔT,

$$R_i = \frac{W(i\,\Delta T + T_0) - W((i-1)\Delta T + T_0)}{\Delta T}. \quad (3)$$

The time counting is started upon the occurrence of the first packet being observed at the *inlet* (sender) or the *outlet* (receiver) of the network [17]. We denote the corresponding start times as T_0^{in} and T_0^{out}, respectively. A packet occurring at time t^{in} at the inlet and at time t^{out} at the outlet contributes its workload to intervals as follows:

$$i^{\text{in/out}} = \left\lceil \frac{t^{\text{in/out}} - T_0^{\text{in/out}}}{\Delta T} \right\rceil. \quad (4)$$

Obviously, the same packet can appear in different intervals at inlet and outlet, resulting in a non-vanishing additional delay at the time scale ΔT as detailed in the next subsection. The throughput time series at the network inlet and outlet are denoted by $\{R_i^{\text{in}}\}_{i=1}^{n}$ and $\{R_i^{\text{out}}\}_{i=1}^{n}$, respectively. They can be obtained from Byte-, bit- or packet counting at each ΔT. Compared to the effort related to tracing each and every timestamp of a packet observed at a point of reference, this approach can be considered as *lightweight*.

C. Comparative Output-Input Analysis and Equivalent Bottleneck

As outlined in Section II.B, the *comparison* of the packet delivery processes at the *outlet* of the network with that at the *inlet*, together with subsequent *analysis* – abbreviated as COIA allows for the quantification of the additional delay T^{add}. COIA builds upon the comparative analysis of the application-level throughput as observed at the network inlet and outlet, along a methodology presented in [15] and in absence of perfect clock synchronization of the sender and receiver. In this paper, we furthermore assume lossless and in-order data delivery.

We have already applied COIA to the analysis of throughput time series and related summary statistics. Amongst others, we have shown a classification of bottleneck behaviour based on the comparison of throughput averages, standard deviations and histograms between inlet and outlet [17, 18]. We have illustrated that if the standard deviation of a throughput time series increased between inlet and outlet, the network in-between acts as a shared bottleneck introducing additional delay. Thus, throughput time series and related summary statistics are lightweight descriptions of the data flow at the burst level. As they capture essential properties of the data flow, they can be used as *Reduced Reference Metrics* (RRM). In particular, RRMs can be exchanged between inlet and outlet (i.e. sender and receiver) in order to allow for runtime classification of bottleneck characteristics. Here, we investigate to which extent COIA based on throughput time series allows for an estimation of the additional delay T^{add}. To this end, we consider the end-to-end path as one *equivalent bottleneck*, whose *content* at the end of interval i is described by [17]:

$$X_i = X_{i-1} + D_i\,\Delta T,\ X_0 = 0. \quad (6)$$

X_i describes the amount of data which is still in transit at the end of interval i. As synchronization happens on the first

packet, there is no content at the beginning of a session, i.e. $X_0 = 0$. D_i is a throughput difference between inlet and outlet, called *drift* and defined as

$$D_i = R_i^{\text{in}} - R_i^{\text{out}}. \quad (7)$$

The drift is a central parameter in fluid flow modelling [6], describing the rate at which the content increases or decreases: $D_i > 0$ means $X_i > X_{i-1}$ and $D_i < 0$ means $X_i < X_{i-1}$ if $X_{i-1} > 0$, while vanishing drift $D_i = 0$ implies a constant content $X_i = X_{i-1}$. If the time series are equal, i.e. $\{R_i^{\text{in}}\}_{i=1}^n = \{R_i^{\text{out}}\}_{i=1}^n$, there is neither drift ($D_i = 0$) nor content ($X_i = 0$), the equivalent bottleneck remains empty and the network is considered to be transparent at time scale ΔT. In the special case $\{R_i^{\text{in}}\}_{i=1}^n = \text{const}$, the variations in $\{R_i^{\text{out}}\}_{i=1}^n$ reflect the variations in D_i and thus of the buffer content of the equivalent bottleneck as such.

The fluid flow model assumes a fluid particle flow of constant intensity during the averaging interval. As long as a packet that was sent in interval *i* is received inside the same interval, its additional delay is invisible. However, as soon as a packet belonging to interval *i* traverses an interval boundary and is received in consecutive interval *i* + 1, its additional delay becomes visible. Thus, the proposed methods rely on intervals bounds in order to quantify how much data from interval *i* is delayed to the consecutive interval(s). Let us illustrate this by assuming $D_i > 0$, $D_{i+1} = -D_i$ and $X_{i-1} = 0$: During interval *i*, less traffic leaves the outlet than what was delivered at the inlet. This particular amount $X_i = D_i \Delta T$ is still in transit on the end of interval *i* and is thus delayed. In the next interval *i*+1, the queue gets empty again ($X_{i+1} = 0$). Intuitively, we estimate an additional delay in the order of ΔT across the equivalent bottleneck.

Consider now one packet sent at a uniformly distributed time during interval *i* and received at a uniformly distributed time during interval $i+1$. We arrive at a triangular delay distribution over $](i-1)\Delta T, (i+1)\Delta T]$ with its median and average at ΔT. The latter is consistent with the above result. However, the uncertainty of our estimation amounts to $\pm\Delta T$, which implies that the choice of the time interval obviously has an impact on the error margins. This will be illustrated in Section IV.

D. Related Work

There exist a number of related work areas attempting to model and estimate an additional delay in a distributed data communication systems. Namely, with regards to modelling efforts, we recognize that equation (6) can be seen as the "fluidified" version of Lindley's recursion formula [19]. It expresses the waiting time at the end of an interval as a function of the waiting time at the beginning of the interval, as well as the number of arrivals and the number of departures during that interval. Based on Lindley's recursion, [20] investigated delay in the network based on a constant link capacity, which we do not assume.

The probably most cited paper in the domain of fluid flow modelling and analysis is [6], providing a closed-form analytical description of the buffer content of a fluid buffer of an unlimited size, fed by homogeneous on-off sources. However, authors of [6] do not model the output process of the equivalent bottleneck; this is rudimentarily done in [21]. In [22] we have presented a simple yet effective analysis of the bit rate distribution arising from the AMS-type equivalent bottleneck and thus describe the effect of the bottleneck in terms of changes of bit rate histograms. The main idea in [22] – deriving information on the bottleneck behaviour from a comparison of bit rate histograms at inlet and outlet of a bottleneck – was demonstrated in our previous work through a measurement study of video conferencing traffic [17] and subsequently a measurement study in mobile networks [18]. Both references implicitly used the COIA method, which we present in a great detail in this paper.

The COIA method builds upon a Little Law [26], which however is very general and estimates only a total end-to-end delay spend by a packet in the system (i.e. including transmission, propagation, queuing and additional delays). The COIA methods use refined versions of Little Law and apply it estimate only additional delay values from a sender or receiver viewpoint. Authors of [27-29] have used Little Law as a base for packets' total delay modelling in their systems; however besides putting unattainable in reality assumptions on their systems, these authors do not focus on different delay views, as we do in our research. Namely, in [27], the authors use the Little Law to estimate total delay packets spend in the system, assuming that the number of nodes in a network, a total number of packets exchanged by nodes and total bandwidth available for each node are known. They estimate delays for video, audio, voice and messaging data packets and then use the estimated delay values in their proposal on traffic priority schemas. Similarly, authors of [28] used Little Law to model MAC-level delay of frames exchanged between nodes connected in WiFi network. They assume exponential frames' inter-arrivals and service times. Authors of [29] model a mean total delay experienced by a frame in a flow, assuming a fair scheduling at mobile MAC layer for different flows and fair-capacity sharing at BSs by a 3G mobile network operator.

III. Estimation Methods for Additional Delays

This section provides methods for estimations of the additional delay T^{add} based on throughput time series at the inlet and outlet of the equivalent bottleneck representing the end-to-end network path. Given the limited communication, processing and storage capabilities of mobile devices, and a requirement of timely estimations, *simplicity* of the estimators is a major point of our concern. In particular, the parameters used for the estimation shall be considered in close time proximity to the current interval in order to make the approach as *stateless* as possible. For interval *i*, the latter constraint limits the scope to X_{i-1}, X_i, R_i^{in}, R_i^{out} and D_i.

The proposed methods implement the COIA principle and thus require the exchange of throughput data (R_i^{in}, R_i^{out} or D_i) or of buffer level X_i between inlet and outlet. This exchange can be realized via a separate in-band or out-of-band control channel. Therefore, we assume that either (a) R_i^{in} can be sent from the sender towards the receiver; or (b) R_i^{out} can be sent from the receiver towards the sender – which in both cases allows to calculate D_i and X_i – or (c) the receiver can observe X_i and then calculate D_i and thus R_i^{in}; or (d) the receiver might estimate the sender's average rate $\mathbf{E}[R^{\text{in}}]$.

A. Sender View Ahead (SVA)

The sender is concerned about whether the data sent in the current interval i has been received without experiencing queuing. This means that at an arbitrary time, there should not be any (bottleneck) content left in the network. If there would be any content left, it would be visible at the end of the current averaging interval i as $X_i > 0$. Seen from the sender point of view, which is actually expecting a throughput of R_i^{in}, this amount $X_i > 0$ is considered to be late by a time of

$$T_i^{\text{add}} = \frac{X_i}{R_i^{\text{in}}}. \quad (8)$$

As in this method the sender considers its "left-over" for the next interval $i + 1$, we call this method *Sender View Ahead*. If the current throughput vanishes, i.e. $R_i^{\text{in}} = 0$, the estimation is undefined.

B. Sender View Backwards (SVB)

In another but similar view, the sender is concerned about the impact of "left-over" data from the most recent interval $i-1$ onto the current interval i, while its current expected throughput is R_i^{in}. The *Sender View Backwards* method for the estimation of additional delay is thus defined as

$$T_i^{\text{add}} = \frac{X_{i-1}}{R_i^{\text{in}}}. \quad (9)$$

For constant sender throughput, SVB produces same estimation series as SVA, however moved by one interval, i.e. $\left\{\hat{T}^{\text{add}}_{\text{SVB},i}\right\}_i = \left\{\hat{T}^{\text{add}}_{\text{SVA},i-1}\right\}_i$. Again, $R_i^{\text{in}} = 0$ leads to an undefined estimation.

C. Receiver View Backwards (RVB)

The receiver is concerned about whether the data received in the current interval i has experienced queuing. Such queuing is seen from a non-empty bottleneck at the end of the previous interval $i - 1$, i.e. $X_{i-1} > 0$. Observing a left-over at the end of the interval $i - 1$, the receiver can estimate the transport time needed for this outstanding data based on the current receiver throughput R_i^{out}. The *Receiver View Backwards* method for the estimation of T^{add} is thus defined as

$$T_i^{\text{add}} = \frac{X_{i-1}}{R_i^{\text{out}}}. \quad (10)$$

Replacing R_i^{out} by the nominal link capacity, this method becomes the one used in [20]. Again, vanishing output $R_i^{\text{out}} = 0$ yields an undefined estimation.

D. Maximum of SVA, SVB and RVB (MAX)

Due to the different points of view and depending on throughput and buffering content values, the methods SVA, SVB and RVB are likely to provide different estimations of the additional delay. A pragmatic approach consists of using the most pessimistic estimation, which is the maximum of the three estimations obtained with SVA, SVB and RVB, given as

$$\hat{T}_i^{\text{add}} = \max\left\{\hat{T}^{\text{add}}_{\text{SVA},i}, \hat{T}^{\text{add}}_{\text{SVB},i}, \hat{T}^{\text{add}}_{\text{RVB},i}\right\}. \quad (11)$$

Undefined values in the argument of the maximum operator are ignored; in the worst case, (11) cannot provide any value.

E. Mean Sender View Ahead (MSVA)

The following method is a variant of SVA method, designed for a receiver that may estimate the sender's average rate $\mathbf{E}[R^{\text{in}}]$ instead of the actual value of R_i^{in} and estimate the bottleneck content X_i. The *Mean Sender View Ahead* method for T^{add} estimation is defined as:

$$T_i^{\text{add}} = \frac{X_i}{\mathbf{E}[R^{\text{in}}]}. \quad (12)$$

F. Illustrative Example 1

In this section we present four different cases of network behaviour, in each of which the bottleneck content increases in interval i and then decreases (i.e. the queue releases) in the subsequent interval $i + 1$. This is the same scenario as outlined in Section II.C, which makes us expect $\hat{T}^{\text{add}} = \Delta T$. We present how the proposed methods estimate T^{add}. We consider the following cases:

i. The sender sends one packet in one interval i, the receiver receives it in the subsequent interval $i + 1$.
ii. Constant receiver rate: The sender sends two packets in the interval i and none in the interval $i + 1$. The receiver receives the first packet in the interval i and the second in the interval $i + 1$.
iii. Constant sender rate: The sender sends the first packet in the interval i and the second one in the interval $i + 1$. The receiver receives both packets in the interval $i + 1$.
iv. Alternating sender and receiver rate: The sender sends two packets in the interval i and one packet in the interval $i + 1$. The receiver receives one packet in the interval i and two packets in the interval $i + 1$.

As we can see from Table I, each method might estimate T^{add} differently based on its view. In case i, SVA, RVB and MAX provide the expected additional delay estimation, while MSVA overestimates due to the fact that the average throughput is much lower than the instantaneous value. Case ii provides a different picture. Now, both SVA and SVB are not accurate, while RVB, MAX and even MSVA provide the expected additional delay estimation. In case iii, however, all methods, besides of RVB, provide that value. Finally, case iv is best estimated by SVB, followed by MSVA.

TABLE I
ADDITIONAL DELAYS ESTIMATED BY DIFFERENT ESTIMATION METHODS FOR THE CONSIDERED NETWORK BEHAVIOUR CASES

Case		SVA	SVB	RVB	MAX	MSVA
i	$\hat{T}_i^{\text{add}}$	ΔT	0	N/A	ΔT	$2\Delta T$
	$\hat{T}_{i+1}^{\text{add}}$	N/A	N/A	ΔT	ΔT	0
ii	$\hat{T}_i^{\text{add}}$	$\frac{1}{2}\Delta T$	0	0	$\frac{1}{2}\Delta T$	ΔT
	$\hat{T}_{i+1}^{\text{add}}$	N/A	N/A	ΔT	ΔT	0
iii	$\hat{T}_i^{\text{add}}$	ΔT	0	N/A	ΔT	ΔT
	$\hat{T}_{i+1}^{\text{add}}$	0	ΔT	$\frac{1}{2}\Delta T$	ΔT	0
iv	$\hat{T}_i^{\text{add}}$	$\frac{1}{2}\Delta T$	0	0	$\frac{1}{2}\Delta T$	$\frac{2}{3}\Delta T$
	$\hat{T}_{i+1}^{\text{add}}$	0	ΔT	$\frac{1}{2}\Delta T$	ΔT	0

As general trends, we see that SVA, SVB and MSVA are not accurate when sender varies its throughput, while RVB is not accurate when receiver varies its throughput. All estimation methods but MSVA are susceptible for vanishing throughput values, while the MAX method helps to yield – reasonably exact – values from any of the estimators. In the subsequent section, we will investigate the performance of the proposed estimators based on real application-data traces.

IV. Validation of the Estimation Methods

In this section, we investigate to which extent the proposed methods are able to estimate T^{add}; we evaluate the methods' accuracy under a variety of network conditions, based on time-synchronized lossless data traces collected along the execution of a m-health application.

A. Setup

The MobiHealth system is a distributed system for real-time monitoring of a mobile patient's vital signs (e.g. ECG, temperature) and his location. A patient is wearing a Body Area Network (BAN), consisting of a sensor-set (adequate to his health condition) and a Mobile Base Unit (MBU). The MBU collects and synchronises the sensor-set data, processes (e.g. filters) it and then sends it to a backend-system (BEsys) located in e.g. a healthcare centre. The BEsys makes data available in real-time (or offline) to e.g. healthcare practitioners. The BAN uses an intra-BAN communication network (e.g. Bluetooth) for data exchange between sensor-set and the MBU, and an extra-BAN network (e.g. UMTS) provided by a wireless network provider for data exchange between the MBU and the BEsys, cf. Fig. 1.

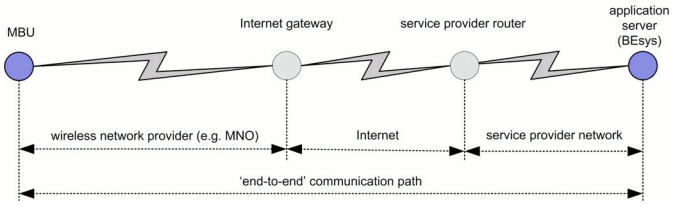

Fig. 1. The MobiHealth extra-BAN end-to-end communication path.

An execution of health telemonitoring application is supported by the proprietary TCP/IP-based *MSP-Interconnect Protocol* (MSP-IP) [23], conforming the Jini Surrogate specification [24]. For detailed description of the MobiHealth system we refer to [7, 8]. For description of our measurements-based performance evaluation methodology and results of previous performance studies we refer to [15, 25].

The QoS requirements posed on the health telemonitoring by healthcare practitioners encompass reliable, error- and loss-free data exchange between the MBU and BEsys at a minimum delay [7]. The use of TCP/IP protocol and the MBU local data storage ensures application data recovery in case of data corruption or loss due to a poor networks performance. Hence the application-data traces are always the loss-less and exhibit in-order delivery. However, this may come at the price of additional delay. Hence, this paper focuses on a minimum data delay requirement posed on the MobiHealth system and methods for the additional delay estimation.

For our studies, we exploit the traces obtained from a health telemonitoring application provided by MobiHealth system to a cardiac patient living in Enschede (the Netherlands), using this application in one location (Twente University). We have used 3G-UMTS network provided by Vodafone NL [15]. Because the network was in a pre-commercial phase at this time, and we were the only users of this network, the delays (and additional delays) observed reflected mainly the impacts of radio channel. In this sense we report upon the 'best case' scenario delay measurements [15].

The application was configured to send different channels of patient's vital signs data from his MBU[1] to the BEsys[2]; resulting in 7, 8, 9, 11 or 12 packets of 524 B per second being sent separately to the network (at inter-packet times of 143, 125, 111, 91 or 83 ms). The protocol stack overhead[3] is 58 B and the overall data rate sent by the MBU to the BEsys is 32.6 to 55.9 kbps. We have also changed the buffer sizes of application and TCP buffer, choosing one of the following combinations: 64/64 kB; 32/64 kB and 32/32 kB. We have synchronized the MBU and the BEsys clocks using NTP. We timestamp each sent and received packet at a time stamp inaccuracy of ± 20 ms [13]. At the MBU, ahead of the TCP socket (i.e. at network inlet) we collect throughput time series $\left\{R_i^{\text{in}}\right\}_{i=1}^{n}$, while $\left\{R_i^{\text{out}}\right\}_{i=1}^{n}$ is collected at the BEsys at the network outlet just behind TCP socket.

Five different data rates, three combinations of buffer sizes, five replications of each experiment and one trace affected by a hardware crash left us with 74 traces on which the validation of our proposed methods is based.

B. Illustrative Example 2

In this section we present an example of the network behaviour derived from the MobiHealth data traces for 8 pps (125 ms inter-packet time), and application and TCP buffer sizes of 32 kB, respectively. In this example, the bottleneck content increases over a period of two seconds and then rapidly decreases (i.e. the queue releases). We investigate how the proposed methods estimate the additional delay. First, in Fig. 2, we present the sender and receiver view on T^{add}. The x-axis displays the time intervals under consideration. Each data point in the figure represents a packet being sent or received, and the y-axis shows its corresponding T^{add} value as derived from the timestamps in the MBU and BEsys data traces.

The packet sent in interval 119 experiences the maximal additional delay of $T^{\text{add}} = 712$ ms, which amounts to almost six nominal inter-packet times (Fig. 2). We can see that this packet is suddenly released at the receiver together with five subsequent packets being queued. This behaviour, which is quite common for mobile links, can be explained as follows: The first packet was corrupted or lost while being in transit and is retransmitted, while the subsequent packets are held until the retransmission of the first packet was successful.

Fig. 3 presents the data rate at inlet (MBU) and outlet (BEsys), and the bottleneck content for intervals as in Fig. 2. The sender has a regular pattern of sending data in four out of five intervals, which matches the ratio between ΔT and the inter-packet time. The reception is however quite bursty with no data being received during intervals 119 to 125. During the interval 126, the receiver gets hold of all six outstanding packets. Afterwards, it continues receiving a data stream during intervals 127 to 129. The content of the equivalent bottleneck shows the increase and a release of the data. Figs. 4 and 5 present estimated additional delay values for the five proposed

[1] Asus laptop (MPIII 1 GHz proc., 640 MB RAM, Win XP OS) using a Nokia 6650 phone as a USB-based modem to 3G-UMTS network

[2] High-performance server placed at the University LAN

[3] Including the MSP-IP (10 B), TCP, IP and PPP overheads

methods. The methods SVA and SVB (Fig. 4) provide increasing estimations of T^{add}, reaching values of 600 ms as they react upon the growing bottleneck content (cf. Fig. 2). SVB with its "backwards" point of view is mostly one interval "late" with its estimations. A vanishing input (e.g. in intervals 114, 119, 124 and 129) makes the results undefined ((7) and (8)). When the bottleneck grows, the RVB method performs very poorly if there is no observed data at the receiver, e.g. in intervals 119 to 125; the result of (9) is undefined. Merely in interval 127, RVB provides T^{add} estimations related to the measured ones. For interval 124, as there is data neither sent nor received, none of the methods SVA, SVB or RVB are able to estimate the additional delay.

The MAX method (Fig. 5) provides estimations of an increasing value, as it uses the maximum value of SVA, SVB and RVB, where especially the first two react upon the growing bottleneck content (cf. Fig. 3). Due to the unavailability of the underlying estimations, the estimation of MAX for interval 124 is undefined.

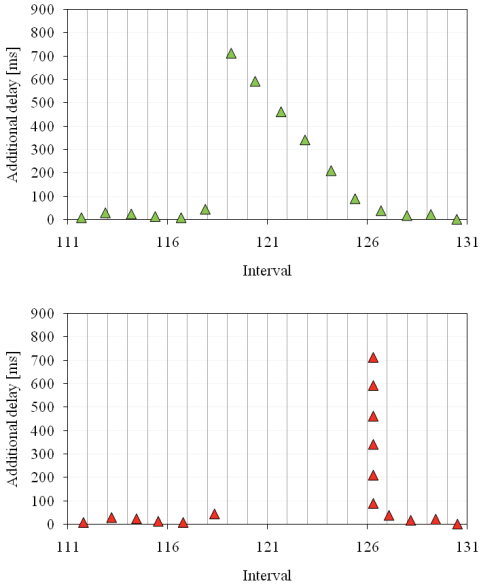

Fig. 2. Sender and receiver packets trace and the corresponding additional delays interval duration $\Delta T = 100$ ms .

The MSVA method delivers quite conservative estimations. It takes the estimated sender behaviour instead of the real one into account. Hence, it assumes that 420 B are sent in each interval of $\Delta T = 100$ ms . The method accounts this data for bottleneck content, which results in an over-estimation of additional delays. Similarly to SVA, this method reacts upon the growing bottleneck content. Just before releasing the queue, MSVA estimates $\hat{T}^{\text{add}}_{\text{MSVA},126} = 819$ ms , which is to be compared to real value of $T^{\text{add}} = 712$ ms and the SVA-based estimation of $\hat{T}^{\text{add}}_{\text{SVA},126} = 600$ ms . In the interval 124, despite the fact that there is no data being sent, MSVA (as the only method) estimates $\hat{T}^{\text{add}}_{\text{MSVA},124} = 619$ ms .

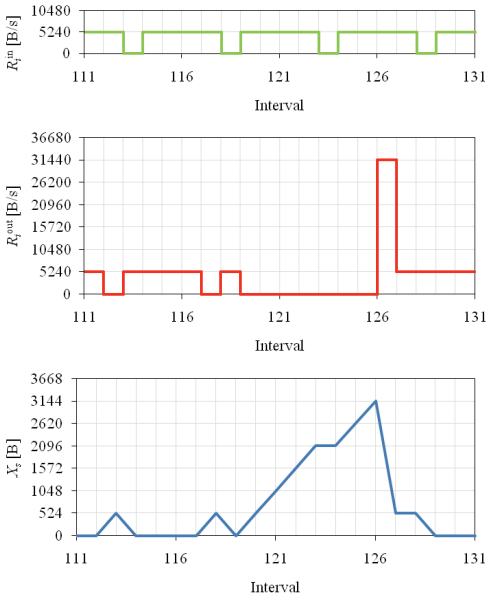

Fig. 3. Data rate at inlet and outlet and content of the equivalent bottleneck.

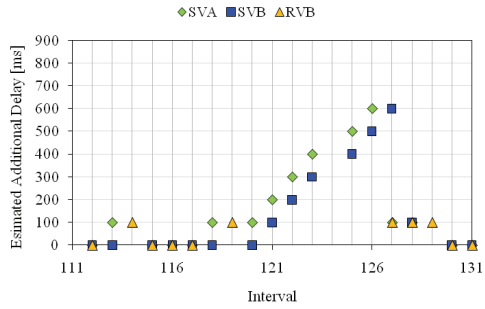

Fig. 4. Estimated additional delays by SVA, SVB and RVB for $\Delta T = 100$ ms .

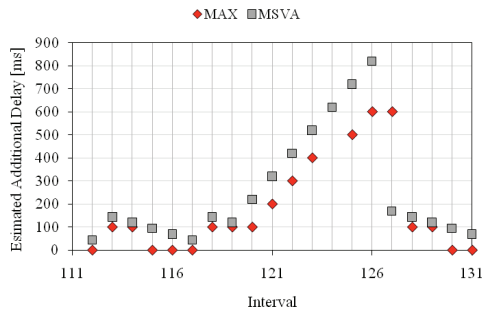

Fig. 5. Estimated additional delays by MAX and MSVA for $\Delta T = 100$ ms .

C. *Accuracy of the Estimated Maximal Delay*

Having examined in details an illustrative example in the previous section, we now present the cumulative results for the accuracy of estimations of additional delays along all 74 application-level traces collected in the MobiHealth system. For each trace, from the sender (MBU) and receiver (BEsys) timestamps, we derive $T^{\min}$ as the minimum delay value ever occurred in that trace. Then, for each packet p of this trace, we derive its (measured) additional delay T_p^{add}. The *maximum additional delay* ever occurred in the trace is denoted as:

$$T_{\max}^{\text{add}} = \max_p \left\{ T_p^{\text{add}} \right\} = \max_p \left\{ t_p^{\text{out}} - t_p^{\text{in}} - T^{\min} \right\}. \quad (13)$$

Then, for each method *M*, we calculate its maximal estimated additional delay in the trace as

$$\hat{T}_{M,\max}^{\text{add}} = \max_i \left\{ \hat{T}_{M,i}^{\text{add}} \right\}. \quad (14)$$

The *relative estimation error* (i.e. relative to the chosen ΔT) is then defined as

$$e_M = \frac{\hat{T}_{M,\max}^{\text{add}} - T_{\max}^{\text{add}}}{\Delta T}. \quad (15)$$

We have observed over all traces that a typical value of $T_{\max}^{\text{add}}$ is in the order of magnitude of 300 ms, with some exceptions reaching up to 712 ms as shown before. Fig. 6 presents the cumulative distribution function (CDF) of the relative estimation error e_M of all five methods SVA, SVB, RVB, MAX and MSVA for $\Delta T = 100$ ms. Fig. 7 applies $\Delta T = 300$ ms and Fig. 8 $\Delta T = 1$ s, respectively. The *x*-axes display the estimation error in % of ΔT.

For $\Delta T = 100$ ms (Fig. 6), the estimation error for SVA ranges from $-1.2\Delta T$ to $1.1\Delta T$, for SVB from $-1.5\Delta T$ to $1.4\Delta T$ and for MAX from $-1.2\Delta T$ to $1.4\Delta T$. In most cases, the error is bounded by $\pm\Delta T$. RVB under-estimates additional delay by $-5\Delta T$ to $-0.5\Delta T$. This is due to its inability to trace growing content in the equivalent bottleneck, if there is no data being received (c.f. Section IV.B). The MSVA displays error values between $-0.3\Delta T$ to $2.2\Delta T$; it has a clear tendency to overestimate the additional delay (c.f. also Section IV.B).

For $\Delta T = 300$ ms (Fig. 7), the estimation error for SVA ranges from $-0.8\Delta T$ to $0.7\Delta T$, for SVB from $-0.9\Delta T$ to $0.8\Delta T$ and for MAX from $-0.9\Delta T$ to $0.8\Delta T$. The majority of absolute errors is found in the range of $-\frac{1}{2}\Delta T ... \frac{1}{4}\Delta T$. RVB exhibits errors in range of $-1.7\Delta T$ to $0.2\Delta T$; it again shows a tendency to under-estimate the additional delay. The MSVA has an error range of $-0.4\Delta T$ to $0.7\Delta T$, with a less pronounced tendency to overestimate the value because $\mathbf{E}[R^{\text{in}}]$ is more close to the actual R_i^{in} values on this time scale. For any of the methods, the relative estimation error has decreased significantly at this time scale, which happens to coincide with the typical value of $T_{\max}^{\text{add}}$ in the order of 300 ms.

For $\Delta T = 1$ s (Fig. 8), the most precise estimation methods are SVA and MSVA, with relative estimation errors ranging from $-0.6\Delta T$ to $0.04\Delta T$ (SVA) and to $0.12\Delta T$ (MSVA). Yet, 99 % of the e_M values for both methods are found in range of −10 % to 4 % (SVA) and to 12 % (MSVA). The reason for such a high accuracy of the latter method is as follows. In the MobiHealth system, the sender is precisely timing the 1s time intervals during which it sends an integer number of packets. Hence, $\mathbf{E}[R^{\text{in}}] = R_i^{\text{in}}$ and that results in a precise estimation of the additional delays. This particular case shows that MSVA method can be beneficial if ΔT corresponds to an integer multiple of the nominal inter-packet time. From Fig. 8, we also conclude that SVB and RVB and hence MAX have a tendency to overestimate the additional delays. The estimation error ranges from $-0.6\Delta T$ to $0.8\Delta T$.

In general, the best estimations are delivered by the SVA method, with a certain tendency to underestimate the additional delay, and by the MSVA method if the averaging interval matches the periodicity interval of the traffic. On the time scale $\Delta T = 100$ ms, which is considerably smaller than the major additional delays, the estimation error is more or less limited by this time scale, with exception of the method RVB. However, we perceive typical errors that are (much) less than the time scale itself. In other words, the methods even allow for observation of additional delays that are smaller than the time scale on which the measurements are carried out.

V. Conclusions and Outlook

In this paper, we propose five methods for the estimation of additional delay induced by radio link problems, queuing phenomena, etc. All methods require lightweight communication of status between sender and receiver; hence we advocate for their use for performance management of applications operational in mobile environments, where wireless access networks provide variable performance.

The methods implement the Comparative Output-Input Analysis (COIA) method, essentially based on the lightweight exchange and comparison of throughput time series between sender and receiver. Using traces from a mobile health telemonitoring system, we show that the most precise estimation methods are based on the sender view on the content of the equivalent bottleneck between two end nodes and the current or average throughput expected by the sender. Furthermore, the pragmatic method that is maximising three observations helps to yield conservative estimations of additional delays, which is of particular interest for safety-critical systems.

The estimation error of all methods depends amongst others on the time scale of the averaging interval for the throughput, which is determined by the capabilities of the terminal in question, and by the characteristic time scales of the events to be discovered. Our results show that "oversampling" in terms of a time resolution finer than the major additional delays to be observed does not necessarily improve the quality of the estimation. Indeed, significant additional delays can be observed and quantified approximately on time scales that are significantly larger than those of the queuing phenomena themselves. This speaks in favour of the methods to be implemented in a constrained mobile environment. Here, the methods can help to quickly discover extraordinary additional delays. Thus, they enable lightweight control loops for adapting an application's data stream to volatile network conditions, e.g. by compressing of suppressing low-priority data.

Future work will address detailed investigations of the capability of the methods to detect arbitrary peaks of additional delay in different application domains. In this context, we will further analyse the impact of the averaging interval, in particular in the context of time scales induced by the application on one hand and enabled by the underlying instrumentation on the other hand. We will also address issues of systematic underestimations of additional delays, undefined estimations, sender jitter, time synchronisation, and clock drifts. Ultimately, we aim at providing design guidelines for delay-aware application protocols optimizing the mobile user's experience while dealing at run-time (and desirably proactively) with the performance provided by the underlying networks. We finally propose an implementation and an evaluation of the proposed methods in the application control mechanisms of the operational MobiHealth system.

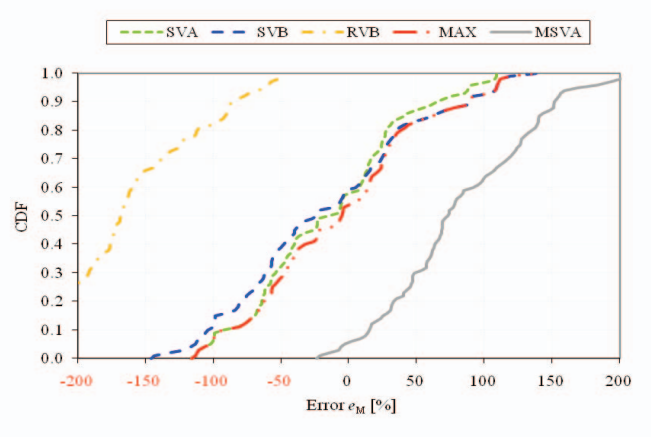

Fig. 6. Empirical CDF of the relative estimation error e_M for $\Delta T = 100$ ms .

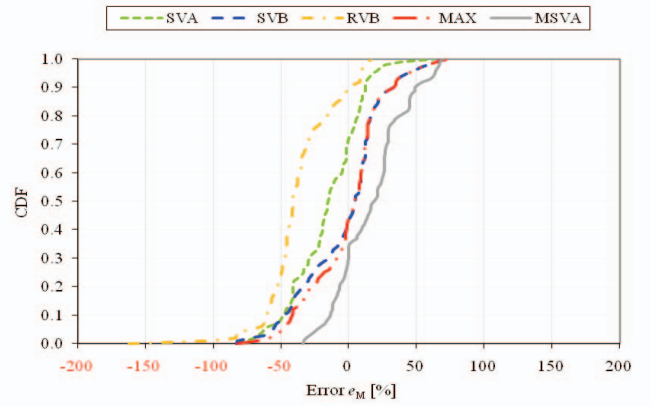

Fig. 7. Empirical CDF of the relative estimation error e_M for $\Delta T = 300$ ms .

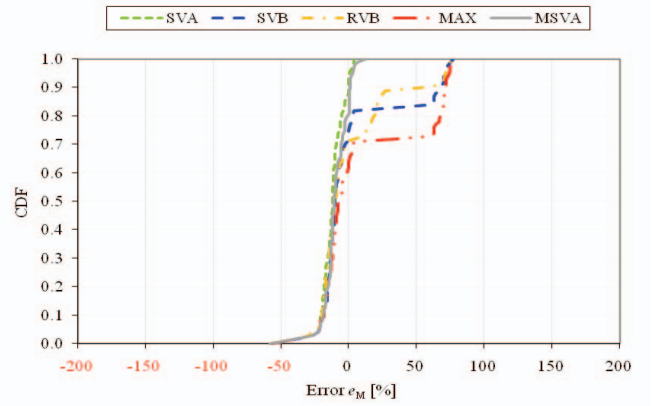

Fig. 8. Empirical CDF of the relative estimation error e_M for $\Delta T = 1$ s .

REFERENCES

[1] U. Hansmann, L. Merk, M. Nicklous, and T. Stober, Pervasive Computing: The Mobile World: Springer, 2003.

[2] ITU-T, "Definitions of terms related to QoS," Rec. E.800: ITU, 2008.

[3] D. Chalmers and M. Sloman, "A survey of QoS in mobile computing environments," IEEE Com. Surveys and Tutorials, (2)2: 2-10, 1999.

[4] L. Zhang, Z. Liu, and C. Xia, "Clock Synchronization Algorithms for Network Measurements," Proc. of INFOCOM, 2002.

[5] L. Kosten. Stochastic theory of a multi-entry buffer. Delft Progress Report, (1): 10–18, 1974.

[6] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data handling system with multiple sources. Bell Syst. Tech. J., 61: 1871–1894, 1982.

[7] A. van Halteren, et al., "Mobile Patient Monitoring: the MobiHealth System," The J on IT in Healthcare, 1(2): 365-373, 2004.

[8] MobiHealth B.V., "Putting care in motion," 2007, visited on 10/01/2009.

[9] S. Tachakra et al., Mobile e-Health: the Unwired Evolution of Telemedicine. Telemedicine J and e-Health, 9(3): 247-257, 2003.

[10] R. Schott, Wearable defibrillator. J of Cardiovascular Nursery, 16(3): 44-52, 2002.

[11] P. Carlsson, D. Constantinescu, A. Popescu, M. Fiedler, and A.A. Nilsson. Delay performance in IP routers. Proc. of HET-NETs, 2004.

[12] J. Nielsen. Usability Engineering. Morgan Kaufman, SF, USA, 1994.

[13] K. Wac, P. Arlos, M. Fiedler, S. Chevul, L. Isaksson, and R. Bults. Accuracy evaluation of application-level performance measurements. Proc. of NGI, pp. 1–5, 2007.

[14] P. Arlos. On the Quality of Computer Network Measurements. Ph.D. Thesis 2005: 05, Blekinge Institute of Technology, School of Engineering, Dept. of Telecommunication Systems, October 2005.

[15] K. Wac and R. Bults, Performance evaluation of a Transport System supporting the MobiHealth BAN ip: Methodology and Assessment, MSc final report, University of Twente, the Netherlands, 2004.

[16] M. Claypool, The Effect of Latency on User Performance in Real-Time Strategy Games. Elsevier Computer Networks, 49(1): 52-70, 2005.

[17] M. Fiedler, K. Tutschku, P. Carlsson, and A.Nilsson. Identification of performance degradation in IP networks using throughput statistics. Proc. of ITC-18, 2003.

[18] M. Fiedler, L. Isaksson, S. Chevul, J. Karlsson, and P. Lindberg. Measurement and analysis of application-perceived throughput via mobile links. The Olga Casals Lecture at HET-NETs, UK, July 2005.

[19] D.V. Lindley, The theory of queues with a single server, Proc. of Cambridge Philos. Soc. (48): 277–289, 1952.

[20]G. Hasslinger, et al., Variability of broadband ADSL traffic: Measurement and time slotted modelling, Proc. of Proc. EuroNGI 2006

[21] S. Aalto. Output from an A-M-S type fluid queue. Proc. of ITC-14, pp. 421–443, 1994.

[22] M. Fiedler and K. Tutschku. Application of the stochastic fluid flow model for bottleneck identification and classification. Proc. of DASD, Orlando, FL, pp. 35-42, 2003.

[23] Dokovsky, N., A. van Halteren, and I. Widya. BANip: Enabling Remote Healthcare Monitoring with Body Area Networks. Proc. of FIJI, 2003.

[24] SUN, The Jini™ Technology Surrogate Architecture Overview, 2001.

[25] Bults, R., et al. Goodput Analysis of 3G wireless networks supporting m-health services. Proc. of IEEE ConTEL05, Zagreb, Croatia, 2005.

[26] Little, J. D. C. A Proof of the Queueing Formula L = λ W Operations Research, 9, 383-387, 1961.

[27] Garetto, M. and Towsley, D. 2003. Modeling, simulation and measurements of queuing delay under long-tail Internet traffic. SIGMETRICS Perform. Eval. Rev. 31 (1): 47-57, 2003.

[28]M. Barry, A. Campbell, A. Veres, Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks, INFOCOM 2001

[30] G. Aniba, S. Aïssa, A General Traffic and Queueing Delay Model for 3G Wireless Packet Networks, Telecomm. and Networking: 123-137, 2004.