Experiments with Reliable Data Delivery in Wireless Sensor Networks

[#] Mihai Marin-Perianu, Paul Havinga
University of Twente, Department of Computer Science
{m.marinperianu, p.j.m.havinga}@ewi.utwente.nl

Abstract

The increasing complexity of Wireless Sensor Networks (WSN) applications require simple, yet reliable, underlying networking mechanisms. In this paper, we describe the experiments performed to establish the real challenges and sources of errors for the reliable data delivery problem. We also discuss several implementation solutions and try to establish the issues that should be taken into account in the design phase. The results are obtained by field measurements, therefore we consider them relevant and useful. Our work relies on tight interaction among transport, routing and medium access layers, with the overall goal of achieving energy efficiency through cross-layer optimizations.

1. INTRODUCTION

Making wireless sensor networks feasible and useful in complex, real-world applications has become one of the most challenging research issues. A large range of business processes can clearly benefit from relocating more logic at the point of action, through the use of intelligent sensor nodes, but we still lack a systematic approach to put this into reality. In this paper we try to move a small step further, by studying the problem of reliable data transport, as this represents a basic component for enabling communication in real applications. Compared to the related work in the area, our efforts have an explicit experimental focus. We try to study which are the practical problems and challenges, in order to derive some useful design guidelines. We argue that simple, end-to-end based solutions should not be neglected, especially when correlated with light optimizations exploiting the wireless, neighbourhood-based communication particularities.

This paper is organized as follows. The following section motivates the importance of reliable data delivery by setting it in the context of business and industrial applications. Section 3 overviews the related work. In Section 4 we describe the steps followed in order to achieve an efficient cross-layer solution and in Section 5 we present the experiments and comment the results. Finally, Section 6 formulates the conclusions.

2. APPLICATION SETTINGS

Transport and logistics are complex processes that can be assisted by WSN technology. The concrete scenario concerns the distribution process of goods from the production plant to the retail stores. During this process, the goods are usually placed on rolling containers, or Returnable Transport Items (RTI). Due to human intervention to this process, errors occur often, e.g. environmental parameters are improper, goods are lost, loaded incorrectly or delivered to the wrong store. By outfitting the RTI with wireless sensor nodes, the quality and efficiency of the distribution process can be improved.

Industrial site security and safety represent another application domain for WSN. Context-aware sensor nodes can check whether the situations comply with certain regulations and warn about possible dangers.

The specific requirements of these applications imply a shift, in terms of functionality, from a static pattern to a dynamic one. The backend application has to be able to access the sensor nodes (or groups of sensor nodes) running certain tasks, to update parameters or to deploy and run new tasks reflecting functional changes. Both updating and task deployment rely on guaranteed delivery of the information because of the strict requirements of business processes. However, achieving reliable data delivery among error-prone sensor nodes is an intricate task. On the one hand, the communication protocols should be ingenious enough to provide best-effort robust communication, even in harsh conditions. On the other hand, the overhead and the additional energy consumption should be kept to a level that still motivates the worthiness of the whole process.

3. RELATED WORK

The problem of reliable data delivery has received increasing concern lately. The efforts have generally focused on highlighting the differences to the traditional transport schemes and designing protocols tailored to the particularities of WSN.

Wan et al. [1] have proposed a hop-by-hop error recovery approach, named Pump Slowly Fetch Quickly (PSFQ), which aims at eliminating error accumulation across multiple hops. The key idea of the protocol is to slowly inject messages into the network, while quickly performing NACK-based repairs from direct neighbours. The main shortcomings concern the energy and memory expenditure by keeping nodes listening for repair requests, the unnecessary delays for low error rate networks and the separation of the transport layer from MAC and routing. For the nodes-to-gateway case, Sankarasubramaniam et al. propose ESRT [2] (Event-to-Sink

Reliable Transport), which focuses on reliable event detection with minimum energy expenditure and congestion resolution. The goal of ESRT is to adjust the reporting frequency of source nodes in order to achieve the desired reliability while keeping energy consumption minimal.

A step further is performed in [3] by defining an information-driven reliability (IDR) mechanism for wireless sensor networks. It is significant to remark that IDR is not a transport protocol, but it brings in a more general discussion regarding architectural choices for implementing reliable data transfer in sensor networks. This discussion is extended in [4] by Stann and Heidemann. They propose RMST (Reliable Multi-Segment Transport), which is a transport layer designed for Directed Diffusion. Its goal is to eventually deliver fragmented data to all subscribing sinks. The receivers (sinks or caching nodes) are again responsible for loss detection. RMST provides no guarantees about latency or delivery order.

Patra et al. [5] use the concept of Delay Tolerant Networking in the design of DTNLite to provide reliable data transfer architecture for sensors. The basic mechanisms proposed are store-and-forward using stable storage and custody transfer. The first is intended to alleviate buffer overflow problems, while the latter achieves reliability by passing transfer responsibility on a hop-by-hop basis. Multiple custody transfer policies have been evaluated, in order to prove that it is possible to optimize for a specific objective like energy or delay while using just local information. However, reliable custody transfer proposed in DTNLite involves stable storage usage, which might result into high energy consumption, long delay penalties for intensive communication conditions.

GARUDA [6] is a solution for reliable sink-to-sensors downstream data delivery, based on the construction of a loss recovery infrastructure called the core. The repairs occur in two phases: first the core nodes recover all lost packets, and then they perform retransmissions for the non-core nodes. The core is built based on hop count information and it approximates the minimum dominating set (MDS). Although this hierarchical structure has efficiency benefits, constructing and maintaining it can add substantial overhead and affect negatively the feasibility of the solution.

Our approach complements the related work in the sense that it aims at establishing the real challenges on an experimental basis. Hence, our field measurements have been carried out in a testbed, taking into account all the difficulties arising in a real implementation. In addition, we describe a complete solution rather than a separate transport layer, thus providing better insight into the practical aspects of the problem.

4. CROSS-LAYER APPROACH

In this section we describe the goals of our experiments and present a cross-layer approach for efficiently solving the reliable data delivery problem. As described in Section 2, the scenario consists of a backend application that needs to update parameters or deploy new tasks on certain nodes within the underlying WSN. From the transport perspective, the problem reduces to guaranteed delivery of a byte-stream, from the gateway connected to the backend application towards the targeted node(s).

It has already been noticed in [4] that the transport layer should not be completely separated from routing and MAC for efficiency reasons. Motivated by experimental results, we support the idea of cross-layer optimization by using information available at the network level. Our work relies on previous experiences with EYES nodes [7,10] and LMAC [8], an energy-efficient, lightweight medium access control protocol. LMAC is based on scheduled access, which means that each node obtains periodically the right of using the medium for a fixed time interval. More specifically, time is divided in frames and each frame consists of a number of slots. By default, frame length is one second and it holds 32 slots, but these values can be adapted to specific requirements. Every node gets to control one time slot in every frame. The time slot is divided into a short part, allocated for the control message (CM), and a longer one, intended for the potential data message (DM) that the node might want to transmit. Nodes always listen for the period of the CM, but, if they are not the intended receivers of the DM, they go to sleep state until the next time slot. Consequently, the nodes save energy by not using their transceivers while not needed and the number of collisions is kept very low compared to CSMA-based protocols.

The most important issue for the data transport problem is represented by the additional functionality that LMAC can provide at network level, namely:

- *Neighbour information*, including neighbour ID, link quality and distance to gateway (in hops);
- *Automatic broadcast propagation*, with broadcast storm prevention through filtering;
- *Node-to-gateway routing*, based on the neighbours hop information;
- *Acknowledgements and retransmissions*, within the local neighbourhood.

All the data needed for providing these functions is exchanged, in a very compact form, within the CM. This directly translates into the advantage of obtaining extra functionality without additional energy costs. Of course, there are costs in terms of memory required for storing neighbour and message information, but, overall, everything indicates a good starting point for implementing an efficient, lightweight transport protocol. In the following, the most important ideas together with the cross-layer optimizations are described step by step.

First, we assumed that, for guaranteed delivery, an end-to-end acknowledgement method has to exist; otherwise the source cannot be sure that the data has been correctly received and thus cannot release it from the memory. The already available node-to-gateway routing, based on neighbour hop count information, constitutes a straightforward solution to this problem.

Second, it occurred obvious to extend the routing with a simple procedure for the reverse case (i.e. gateway-to-node). We achieved this by using the first transmission packet, which is anyway usually used for setting up transport session, sequence numbers, etc. The exact procedure is as follows: (1) the first packet is broadcasted in the network until it reaches the receiver, which (2) processes and acknowledges it using the node-to-gateway path; (3) on the way to the gateway, each intermediary node stores the neighbour from which it has received the acknowledgement as the next hop for the gateway-to-node path. The following data packets will be routed according to this path. In the case a node from the path crashes, the process has to be repeated by re-broadcasting the current packet and establishing a new path.

Third, we had to consider the acknowledgements and retransmission scheme provided by LMAC, which is an optional feature that enables nodes to determine whether their packets have been received correctly. This is done by reading the acknowledgement field from the CM of each time slot, so it has two major advantages: it does not consume additional energy and broadcast messages can also be acknowledged by all the recipients. If the expected acknowledgement is not received, the sender automatically retransmits the packet for a maximum of four times before giving up. It is well known that the communication medium in WSN is extremely error prone. However, from our experiments, it turned out that, for quasi-static settings, where the connectivity is maintained, this simple mechanism might suffice. The main cause of errors was represented by losing connectivity to the network due to mobility or hardware problems. We tested two techniques for dealing with these situations: end-to-end retransmissions and local repairs from cache. The next section will provide a detailed comparison between the two alternatives.

Finally, we developed a simple mechanism for practically enabling the cross-layer interaction. We extended LMAC with control points for the three important operations we were concerned with: packet incoming, packet outgoing and acknowledgement. The transport protocol can register callback functions through which it can take control over the normal operation. In this way, we can combine functionality from MAC, routing and transport layers, and still maintain a uniform interfacing method.

5. EXPERIMENTS

This section presents practical tests concerning reliable data delivery from the backend application, through the gateway, to a single sensor node, specified by a unique ID. This is the very basic method that enables task deployment and parameter updating; yet we are not aware of any complete solution to this problem. In the following we briefly describe the hardware platform used in experiments and the operating system support, then we detail the various tests performed and comment the results.

A. Hardware platform

The tests have been performed using an improved version of the sensor node platform developed for the EYES project [7,10]. The onboard micro-controller is the Texas Instruments MSP430f169. The node is equipped with a Nordic nRF905 multichannel radio transceiver for wireless communication. The radio operates in the 868MHz and 915MHz band and has a maximum data rate of 100kbps. Other features on the device are a 2Mbit serial flash memory for secondary storage and a RS232 interface for communicating with other computers.

B. Operating system

The sensor nodes run DCOS [9] (Data Centric Operating System), which is a real-time multitasking operating system designed for enabling data centric architectures. The key points of DCOS are:

- *Real-Time Scheduler*, also providing mutual exclusion for resource sharing;
- *Data Manager*, based on a publish/subscribe mechanism for inter-task communication;
- *Dynamic Loadable Modules (DLM)*, which support system reconfiguration at runtime.

Implementing the cross-layer design on top of DCOS becomes straightforward. The reliable transport layer can be seen as a separate DLM, that can be activated at application request and interacts with the MAC layer through the publish/subscribe mechanism. This has clear benefits over other schemes, because it stimulates energy efficiency through tight layer cooperation and, at the same time, maintains a standard interfacing mode.

C. Three alternative solutions

We have studied in practice three possible approaches bearing increasing complexity:

1) End-to-end acknowledgement for *every packet*;
2) End-to-end, *window-based* acknowledgement;
3) Same as 2, plus *intermediary caching*.

We have therefore started with a simple protocol and gradually added a number of improvements. We believe there is a gap between the existing theoretical models and the implementation conditions, which characterize by frequent errors and unreliable hardware. Lightweight, simple solutions should be hence considered and, as a next step, the tradeoffs of adding various optimizations should be carefully examined.

Due to the guaranteed delivery required by the application, we have considered compulsory to have an end-to-end form of acknowledgement. The important issue was to determine the ways of keeping it minimal without losing reliability. In order to have a basis for our comparisons, we have firstly implemented a straightforward protocol that requires from the receiver an acknowledgement of each packet. This protocol is very simple and robust, yet, as expected, highly inefficient due to the considerable overhead.

The second solution we have explored relies on window-based ACKs combined with selective NACKs. More specifically, the sender transmits the packets within the current window, and then waits for either a complete ACK or a selective NACK indicating the missing packets. The selective NACK packet contains the binary status of the

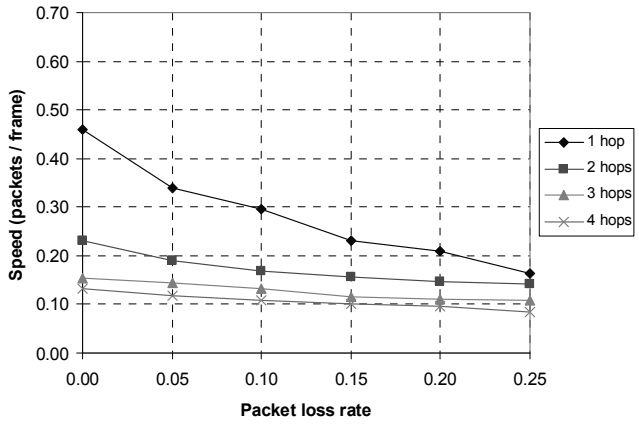

Fig. 1: Protocol 1 - Average speed
(end-to-end ACK for every packet)

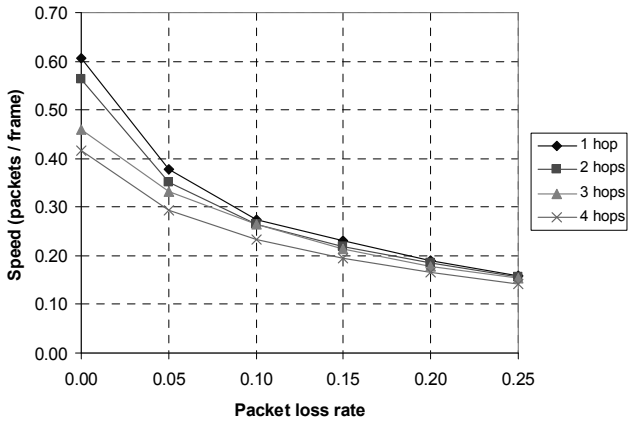

Fig. 2: Protocol 2 - Average speed
(selective, window-based retransmission)

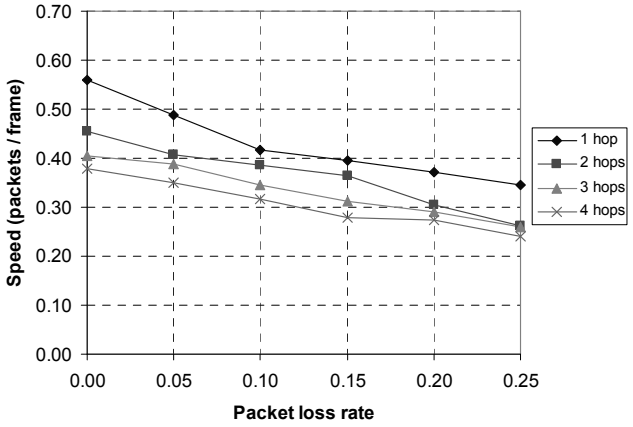

Fig. 3: Protocol 3 - Average speed
(caching on intermediary nodes)

window at the receiver side; for instance "11011" means that the third packet was not correctly received. In this case, the sender can advance its window by two, retransmit the packet, then skip the next two, etc. Sending the binary status information occurs at insignificant cost, as the MAC layer has a fixed time slot reserved for packet transmission, but it has impact on the efficiency by preventing unnecessary duplicates. This solution performs much better in terms of efficiency than the previous one, but it introduces the problem of optimally choosing values for timers and window size. Moreover, in the presence of errors, the average transmission speed (and proportionally, the efficiency) drops severely.

Finally, motivated by the acknowledgement and retransmission scheme of LMAC, we have extended the window-based version with error control on the intermediary nodes. Every node in the routing path maintains a small cache, equal with the window size, from which it periodically sends one packet to its next hop neighbour. It advances to the next packet only when notified of correct LMAC acknowledgement through the callback function (see Section 4). If an intermediary node crashes, the previous node in the path notices that one of its neighbours is no longer active (again, due to the information provided by LMAC) and broadcasts the current packet. The receiver will reply with an ACK that will determine the creation of a new path and the transmission can continue. This protocol assures better performance in case of errors, as the repairs are done locally, but this comes at the cost of consuming memory for caching packets.

D. Tests and results

For each protocol we have performed numerous tests, in various physical settings, so that to collect an extensive range of data. A typical experimental run consists in transmitting an application message (usually of 400 bytes) from a PC, through a serial link, to a gateway node and further to the intended receiver node. The transmission is fully streamed, so the gateway node does not have to maintain any cache. Moreover, the communication on the serial link is done reliably. The sender starts by sending an initial announcement packet, used for setting up the session, the sequence numbers and the routing path, then it progressively sends the message, divided in fixed size packets, and waits for acknowledgements according to its protocol scheme.

Besides the correctness and the reliability of the protocols, we have observed two parameters for characterizing the performance and the efficiency: *transmission speed* and *goodput*. We have chosen to express the transmission speed in the percentage of a data packet sent per frame, as it gives a correct view on the performance regardless of specific settings. Indeed, LMAC can support different packet sizes and frame lengths, in order to cope with certain requirements, such as node density or radio link quality. Throughout our experiments, the frame length has always been one second and the usual packet sizes have been 32, 64 and 128 bytes. The second parameter, goodput, represents the amount of useful information successfully transmitted, scaled to the total communicated data (including headers, acknowledgements, retransmissions, etc.). It offers a relevant indication on the overhead and the energy efficiency of the protocol.

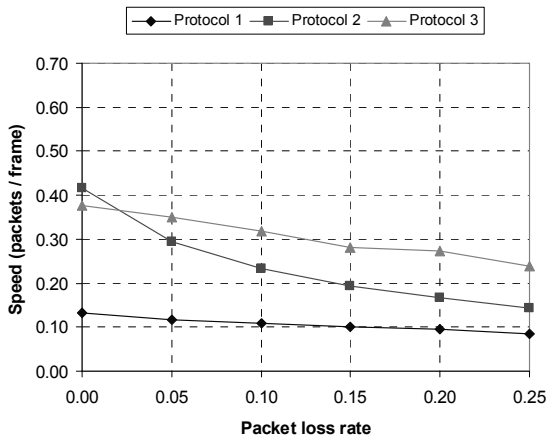

Fig. 4: Comparison of average speed for 4 hops.

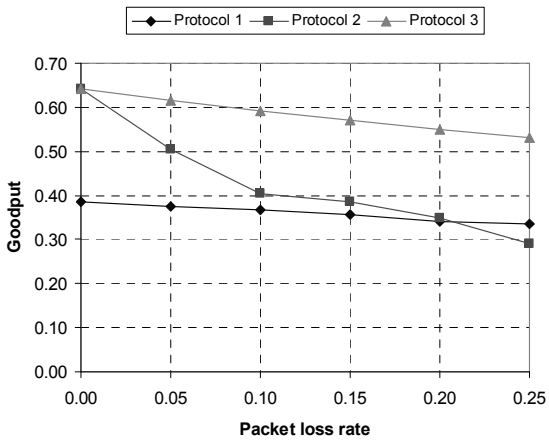

Fig. 5: Comparison of goodput for 4 hops.

The experiments have been performed so as to study the behaviour of transmission speed and goodput in relation to two variable parameters: *packet loss rate* and *hop distance*.

For versions 1 and 2, the packet loss rate is determined at endpoints, whereas for version 3, all the intermediary nodes count the number of local errors and, after message completion, they report back to the application. The hop distance depends on the particular physical deployment of the nodes and is known before starting the transmission.

The graphics from figures 1, 2 and 3 depict the average transmission speed as a function of the packet loss rate, for hop distances between one and four. It can be clearly noticed that the first protocol exhibits substantial overhead and its performance decreases drastically for multi-hop situations. This solution should be applied only when there are strict limitations on the size and complexity of the code that can be run on the nodes.

The second protocol performs much better, especially for low error rates (below 10%), as it requires fewer acknowledgements and permits the receiver to request retransmissions only for the lost packets from the current window. The major difficulty with respect to implementation consisted in finding the proper values for the window size and the timeout timers. A larger window means fewer ACK/NACK packets, but also longer delays in case of errors. Both figures 2 and 3 represent results considering a window size of 5 packets, which experimentally proved to be an acceptable trade-off.

Figure 3 shows that local caches and retransmissions attenuate the effect of communication errors. However, this comes at the cost of reserving a small amount of memory (equal with the window size, in our tests) on each intermediary node. For low error rates, we can notice a small additional latency compared to the second protocol. This is explained by the fact that the nodes have to store the incoming packets and schedule them for sending, therefore their time slot is occasionally ended before the actual transmission.

Eventually, the graphics from figures 4 and 5 provide a comparative view on performance (average speed) and efficiency (goodput), respectively, within a multihop environment. It is interesting to notice that, from the speed perspective, the third protocol is not always the best choice.

Indeed, for error rates less than 5%, the second protocol exhibits almost similar performance. Besides, it has the advantage of not consuming extra memory for caching. Nevertheless, in terms of goodput, performing local repairs reduces the number of ACK/NACK packets at the transport layer and therefore the third protocol proves to be the most efficient.

E. Errors

We conclude this section with a brief discussion on the type and quantity of communication errors encountered throughout the experiments. All the tests have indicated that errors most usually occur in burst and thus determine the loss of one or more packets. Possible reasons include losing connectivity to a neighbour from the routing path, radio interferences and hardware problems. For this reason, we have described the performance and efficiency of protocols in relation to packet loss rate, rather than bit error ratio (BER). The detection of bit error rates is carried out at MAC level, through CRC codes, and it determines automatic retransmission of the affected packet.

The last test performed aims to give a quantitative image about the loss rate. Therefore, we have continuously executed numerous transmissions using protocol 3, over five hops, and collected the data in Fig. 6. It appears obvious that communication errors generally determine burst losses, which in turn affect the overall transmission speed. Another important result is the average loss rate, which has a value of 3,323% (also plotted in Fig. 6). This relatively low value clearly supports the usage of simple protocols.

6. CONCLUSION

There are a number of relevant remarks that derive from the above presented experiments. Starting from the evident need for reliable transport mechanisms, it is important to establish the complexity they must achieve for coping with real environments. Moreover, we argue that transport solutions

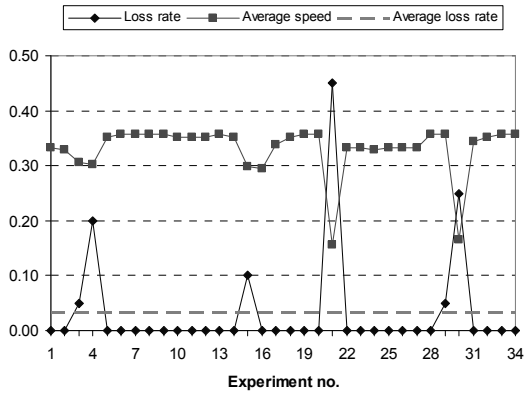

Fig. 6: Packet loss rate and average speed
for sending a total of 14 kB over 5 hops, using protocol 3.

should not be separated from MAC and routing layers for efficiency reasons, yet the interfaces should be kept enough uniform and general.

We have described such a cross-layer approach based on DCOS and LMAC. It shows how reliable transport can benefit from the increasing reliability of MAC protocols and from the rich functionality of the operating system.

We have tested three possible techniques bearing increasing complexity and provided comparisons among them. Table 1 summarizes implementation details such as code and data memory footprints both for endpoints and intermediary nodes. The results plead for not neglecting the end-to-end approach, but to correlate it with very simple optimizations exploiting the intrinsic properties of WSN communication. Therefore, careful thought should be given to the complexity of the tasks allocated to the intermediary nodes involved in the reliable transport process. The distributed environment and the undependable nature of sensor nodes make the theoretical assumptions infeasible in practice and raise intricate difficulties at implementation time.

For future work we consider adding flow control support, particularly for protocol 3, which can work with larger windows without experiencing abrupt performance drops. Another possible improvement we want to explore is to lessen the memory consumption by distributing the caching probabilistically among the nodes from the routing path. Finally, we intend to study the extension of the problem to the practical case of communicating with groups of nodes collaborating for a common task.

TABLE 1: CODE AND DATA MEMORY FOOTPRINTS

Protocol	Node type	Code [bytes]	Data [bytes]
Protocol 1	Intermediary	716	8
	Endpoint	880	20
Protocol 2	Intermediary	716	8
	Endpoint	1034	34
Protocol 3	Intermediary	1038	282
	Endpoint	1034	34

REFERENCES

[1] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A reliable transport protocol for wireless sensor networks", In *WSNA '02*, pages 1–11, ACM Press, 2002.

[2] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "ESRT: Event to sink reliable transport in wireless sensor networks", In *MobiHoc '03*, pages 177–188. ACM Press, 2003.

[3] T. Stathopoulos and D. Estrin, "An information-driven reliability mechanism for wireless sensor networks", CENS Technical Report 16, UCLA, June 2003.

[4] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks", In *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*, pages 102–112, 2003.

[5] R. Patra and S. Nedevschi, "DTNLite: A reliable data transfer architecture for sensor networks", CS294-1: Deeply Embedded Networks, 2003.

[6] S.-J. Park, R. Vedantham, R. Sivakumar and I. F. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks", In *MobiHoc '04*, pages 78-89, ACM Press, 2004.

[7] Energy-Efficient Sensor Networks (EYES), http://eyes.eu.org.

[8] L. V. Hoesel and P. J. Havinga, "A lightweight medium access protocol (LMAC) for wireless sensor networks", In *INSS 2004*, Japan, 2004.

[9] T.J. Hofmeijer, S.O. Dulman, P.G. Jansen and P.J. Havinga, "DCOS, a Real-Time Light-Weight Data Centric Operating System", In *ACST 2004*.

[10] S.Dulman, L.van Hoesel, P.Havinga and P. Jansen, "Data Centric Architecture for Wireless Sensor Networks", ProRISC Workshop, Netherlands, 2003.