

# DTSN – Distributed Transport for Sensor Networks

Bruno Marchi<sup>1</sup>, António Grilo<sup>1,2</sup>, Mário Nunes<sup>1,2</sup>

<sup>1</sup> INESC, Rua Alves Redol, N° 9,  
1000-029 Lisboa, Portugal

<sup>2</sup> IST/UTL, Av. Rovisco Pais,  
1096 Lisboa, Portugal

{bruno.marchi, antonio.grilo, mario.nunes}@inesc.pt

**Abstract:** This paper presents the Distributed Transport for Sensor Networks (DTSN), a novel reliable transport protocol for convergecast and unicast communications in Wireless Sensor Networks (WSN)s. In DTSN, the source completely controls the loss recovery process in order to minimize the overhead associated with control and data packets. The basic loss recovery algorithm is based on Selective Repeat ARQ, employing both positive (ACK) and negative (NACK) delivery confirmation. Consequently, DTSN is able to detect when all packets of a session are lost, besides scattered gaps in the data packet sequence. Caching at intermediate nodes is used to avoid the inefficiency of the strictly end-to-end transport reliability TCP-like model, commonly employed in broadband networks. Reliability differentiation is achieved by means of the smart integration of partial buffering at the source, integrated with erasure coding and caching at intermediate nodes. The simulation results attest the effectiveness of both the total reliability and the reliability differentiation mechanisms in DTSN.

**Keywords:** Wireless Sensor Networks, Convergecast, Reliable Transport, Differentiated Reliability, Energy-Efficiency.

## 1 Introduction

Wireless Sensor Networks (WSN)s are highly distributed self-organized systems that rely on significant numbers of scattered low-cost tiny devices featuring strong limitations in terms of processing, memory, communications and energy capabilities, in order to collect some measurements of interest over a given space, making them available to external systems and networks at special nodes designated sink nodes. In order to maximize the autonomy of individual nodes (and consequently the longevity of the network), power saving techniques are commonly implemented causing nodes to sleep most of the time, complemented with low power communications that usually lead to multihop data transmission from sensor nodes to sink nodes and vice versa. While link reliability mechanisms (e.g. MAC layer automatic repeat request – ARQ) can significantly reduce the end-to-end packet loss ratio, some critical WSN applications (e.g., battlefield surveillance applications, intrusion detection applications, or detection and tracking of chemical, biological, radiological, nuclear and explosive agents, etc.) require high or even total end-to-end reliability, demanding the use of a reliable transport layer protocol. On the other hand, some of these applications require *packet-driven reliability* (all packets sent by the source must reach the destination) while others only require *event-driven reliability* (the event must be detected, but it is enough that one notification message reaches the sink node).

Due to these limitations, proven transport protocols like TCP [1], designed to support user applications in infrastructure networks, usually present significant inefficiencies when employed without considerable modification in WSN systems. One of the main factors for TCP inadequacy is related with its strictly end-to-end reliability model, forcing all confirmations and retransmissions to follow the complete path between source and destination, with the consequent toll on the already scarce bandwidth and energy. Several proposals were made for alternative transport protocols, usually focused on specific optimization aspects and/or application scenarios.

WSN transport protocols can be broadly classified in two main functions [2]: reliability and congestion control. Reliable transport protocols can be further subdivided into upstream (mostly unicast/convergecast transmission to aggregator or sink nodes) and downstream (mostly multicast/broadcast of code or configuration updates from sink nodes to sensor nodes). This paper particularly focuses unicast/convergecast reliability, typical of upstream communications. A novel transport protocol called Distributed Transport for Sensor Networks (DTSN) is proposed with the following main goals:

- *Energy-efficiency*: to avoid useless wasting of energy resources through minimization of the control and retransmission overhead;
- *Reliability Differentiation*: to support different reliability grades in order to suit the requirements of different applications regarding throughput, latency and energy consumption. In addition, the support of both *packet-driven* and *event-driven* reliability.
- *Distributed Functionality*: to exploit the WSN resources in cooperative way, so that overall WSN operation is not hindered by the limited capacities of individual nodes.

The rest of the paper is structured as follows. Section 2 presents the most relevant related work; section 3 describes the DTSN protocol; section 4 presents the simulation results; section 5 concludes the paper and lists some topics of ongoing and future work.

## 2 Related work

This section only presents the WSN transport protocols that motivated or influenced more closely the DTSN specification. The reader is referred to [2] for a more complete survey.

Reliable Multi-Segment Transport (RMST) [3] was designed to work on top of the Directed Diffusion [4] to provide reliability in sensor to sink communications (upstream reliability). It offers two simple services: data segmentation/reassembly and guaranteed delivery. A data packet coming from the Application layer is identified by an ID before segmentation. All the resultant segments are characterized by that ID and a fragment number, required to reconstruct the correct sequence at destination. In RMST the receiver is responsible for detecting losses and for triggering the recovery of the missing segments through the generation of NACKs. A receiver is not necessarily a sink, since RMST can be executed in caching mode. In this case intermediate nodes store data traffic and construct the map of the received segments to detect losses. When there is a gap in the sequence of fragments, the receiver sends the NACK after a pre-defined timeout value. In caching mode, the NACKs are

processed at every node in the path from receiver to sender. If some intermediate node finds the missing segment in its cache, it suppresses the NACK and retransmits the segment to the sink. Otherwise the NACK is relayed to the next node toward the source. RMST is affected by various weaknesses that make it somewhat unreliable and inefficient. First of all, there is no full guarantee of delivery. Since the protocol is connectionless and the losses are detected by receivers via timeout, when none of the segments of a data packet is received by the destination, the loss can not be detected because the sink is not aware of the transmission and there is no positive ACK at the end of the transmission. The transport service is standard for all the applications that require some degree of reliability, introducing unnecessary overhead for those packets that do not require total reliability. Finally, the caching mode functionality should be revisited because the timer-based loss detection at the receiver potentially causes early generation of NACKs, contributing to increase the traffic load and to waste energy resources.

Event to Sink Reliable Transport (ESRT) [5] is an application specific protocol designed for event tracking, seeking a balance between reliability and congestion control. It supports communications with a partial reliable grade in upstream. The application layer calculates the minimum number of packets needed to detect an event with a certain reliability grade ( $r$ ). At the sink, given this number and the number of packets effectively received within a defined decision interval ( $\tau$ ), a reporting frequency  $f$  is calculated. The frequency  $f$  is broadcasted to indicate to the WSN nodes the rate at which they should generate packets so that events are reliably detected at the sink. The value of  $f$  is periodically updated based on the observed arrival rate of packets at the sink following a strategy that is primarily focused on reliability rather than energy consumption. However, ESRT does not provide true packet level reliability, since it simply relies on some degree of redundancy to achieve an acceptable probability of event detection. The approach is not energy-efficient and is also not suitable to applications that require full end-to-end delivery guarantees. In ESRT, WSN nodes just perform sensing functionalities and in-network processing mechanisms like caching, data fusion or aggregation are not considered. ESRT also assumes that the channel characteristics are uniform, i.e. all the sensors get the same error rate, irrespective of their location. This is not a realistic assumption and thus the reporting frequency  $f$  will be often overestimated, causing the generation of extra traffic that contributes to congestion. Finally, since reporting frequency  $f$  is broadcasted by means of a long range high power transmission, it may interfere with the upstream flows, increasing the packet loss ratio.

Distributed TCP Caching (DTC) [6] is a TCP enhancement for WSNs, which seeks to take TCP/IP into the WSNs while improving retransmission efficiency through header compression and the use of caching at selected nodes in the path from source to destination, thus minimizing end-to-end retransmissions. DTC is fully compatible with TCP, leaving the endpoints of communication unchanged, only requiring changes at the intermediate nodes. Cross-layer interactions with a MAC that supports positive-ACK helps to infer when packets were lost at the next hop. Intermediate nodes cache the intercepted TCP segment with the highest sequence number seen, and additionally those that may have been lost during the forwarding attempt. The latter remain locked in cache and cannot be overwritten by segments with higher sequence number. The TCP positive-ACKs are intercepted and used by intermediate nodes to infer packet losses or to free cache entries of successfully delivered packets. The next packet expected by the receiver is immediately retransmitted if present in the cache, and the respective

positive-ACK is eliminated. The process continues until the receiver acknowledges all packets transmitted by the source (which means that there are no failed TCP segments in the cache of intermediate nodes), in which case the positive-ACK is allowed to reach the source. In addition, DTC is able to make use of the TCP selective ACK (SACK) option to improve the efficiency of the recovery process. Just like the original TCP, a good estimate of the round-trip-time (RTT) is essential to maximize the performance of DTC. DTC uses a technique called *flying start* that provides a first estimate of the RTT based on the measured connection setup time, achieving 10%-25% of improvement over the *slow start* mechanism in conventional TCP. DTC is able to extend the TCP/IP service to the WSN, facilitating interworking with external IP-based networks. The use of caching significantly improves the efficiency of packet loss recovery, minimizing the energy spent with retransmissions. However, like in TCP, DTC features an inefficiency associated with the transmission of unnecessary positive-ACKs, which is totally controlled by the receiver.

Asymmetric Reliable Transport (ART) [7] addresses the reliability of event notifications and queries, being thus a bi-directional transport protocol. An energy-aware classification of sensors is accomplished so that group of sensors, designated essential nodes (E-Nodes), are selected to cover the entire sensible terrain, using residual energy as the classification criteria. The reception of query requests by all the essential nodes is considered the basis of query reliability. Congestion control is achieved by varying the number of active non-essential nodes (thus limiting the number of issued event notifications). The asymmetry of the protocol resides on the way it treats query (downstream) and event (upstream) reliability. Since queries are essentially multicast, a NACK mechanism is used for query loss recovery, thus preventing ACK explosions. The loss of query messages is detected by means of sequence numbers. A NACK message is issued by an E-Node to the sink whenever a gap is detected. A Poll/Final (P/F) bit is used to distinguish the last query message of a sequence. The reception of a query with P/F set is the only situation when the receiver issues a positive ACK. For event reliability, a positive ACK mechanism is used instead, but the number of ACKs issued by the sink is limited through the use of an event-alarm bit in the notification message. The sink only issues ACKs when the event notifications have this bit set (typically the first message in a sequence of event notifications), thus providing event full reliability, with only partial reliability for individual event notification messages. Although ART tries to take energy efficiency into account in the definition of the E-Node set, the way it deals with reliability is essentially end-to-end, raising the issue of scalability in applications that require high density or redundant sensor reporting (thus requiring a substantial number of E-Nodes). The fact that congestion control is centralized necessarily imposes an extra burden on network resources for the exchange of management/signaling traffic.

### **3 DTSN**

Since not every WSN application requires all the features supported by DTSN, the DTSN specification can be divided into a total reliability service, and a differentiated reliability service. The latter is based on the former and is able to support several reliability grades based on the integration of partial buffering at the source, intermediate node caching and erasure coding.

### 3.1 DTSN Total Reliability Service

This service was thought for critical data transfer requiring end-to-end total reliability. Besides providing total reliability, this service tries to minimize energy consumption, increasing network life-time. Total reliability is achieved by a Selective Repeat Automatic Repeat Request (ARQ) mechanism that uses both negative acknowledgement (NACK) and positive acknowledgement (ACK) control packets. DTSN is compatible with mechanisms of segmentation/reassembly of the high layer data units (e.g., to support acoustic or still image sensing), however, it relegates the segmentation implementation to the higher layers and the size of a packet is application dependent. Soft requirements of routing path stability and bi-directionality are placed on the routing layer in order to leverage the intermediate node caching mechanism of DTSN for performance improvement. Although the following description assumes individual node addressing, DTSN could be adapted to operate on top of more data-driven architectures like Directed Diffusion, in which source-sink relationships are identified by other means. In DTSN, a session is a source/destination relationship univocally identified by the tuple  $\langle \text{source address}, \text{destination address}, \text{application identifier}, \text{session number} \rangle$ , designated the *session identifier*. The session is soft-state by nature both at the source and the destination, being created when the first packet is processed and terminated upon the expiration of an activity timer (provided that no activity is detected and there are no pending delivery confirmations. A randomly chosen *session number* is appended in order to unambiguously distinguish between successive sessions sharing the endpoint addresses and *application identifier*. Within a session, packets are sequentially numbered. The Acknowledgement Window (AW) is defined as the number of packets that the source transmits before generating a confirmation request (Explicit Acknowledgement Request – EAR), and its size depends on the specific scenario. The AW might have a fixed size equal to the minimum value between the output buffer and the size of the application data block to be transferred, which would suite applications requiring low data rate with low energy consumption (less control packets). If, on the other hand, there are throughput requirements, the AW might be set to small values (e.g., half of the output buffer size), so that the source would be able to free output buffer space more frequently and the destination could recover lost segments more quickly. The control of delivery confirmation is done at the source to allow the definition of the trade-off between overhead and speed of loss recovery to be application-specific.

The DTSN session management algorithm at the source is depicted in Figure 1. The source transmits each packet coming from the higher layers, storing it in the output buffer, so that it can be retransmitted latter if required. Upon the transmission of a number of packets equal to the AW size, when the output buffer is full, or when the higher layer protocols have not sent any data during a predefined timeout period, the source requests a delivery confirmation message from the destination by means of an EAR. This may take the form of a bit flag piggybacked in the last data packet (e.g., confirmation request due to the AW size being reached or the output buffer becoming full) or an independent packet (e.g., confirmation request due to the expiration of the EAR timer). Each time a confirmation message (either ACK or NACK) is received, the source frees the output buffer entries whose delivery is confirmed. The reception of an ACK means that there are no gaps in the sequence of packets sent before the respective EAR. On the other hand, a NACK includes a list of gaps in the sequence of received packets, as well as

the next expected sequence number. Its reception causes the source to retransmit the data packets that were not delivered successfully, this time with the retransmission (RTx) flag activated. An EAR is sent after retransmission, which may be piggybacked in the last of the retransmitted packets.

After sending an EAR, the source launches an EAR timer. If the EAR timer expires before an ACK/NACK is received, the source retransmits the EAR packet. The length of the EAR timer is dynamically set taking into account the round-trip-time (RTT), which is estimated with each sample according to the algorithm employed in TCP [8]<sup>1</sup>:

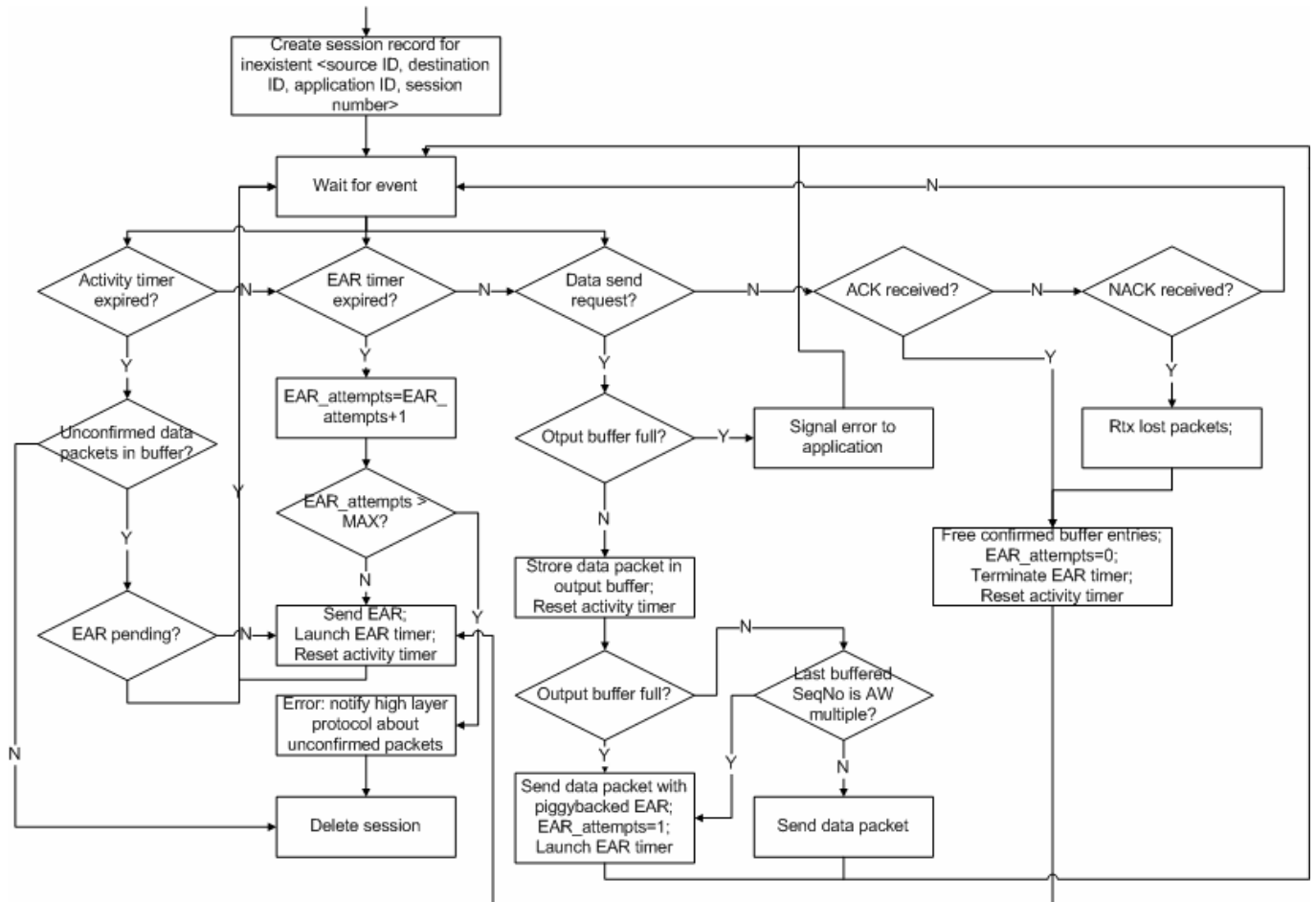
$$EAR\_Timeout = \mu + 4 \times \sigma , \quad (1)$$

Where  $\mu$  is the estimated RTT average and  $\sigma$  the estimated RTT variance:

$$\mu = (1 - \alpha) \cdot \mu + \alpha \cdot x \quad (2)$$

$$\sigma = (1 - \beta) \cdot \sigma + \beta \cdot |x - \mu| , \quad (3)$$

Where  $\alpha < 1$ ,  $\beta < 1$  and  $x$  is the value of the last RTT sample. In this way, throughput degradation is minimized. The number of EAR retransmission attempts is a system configuration parameter.

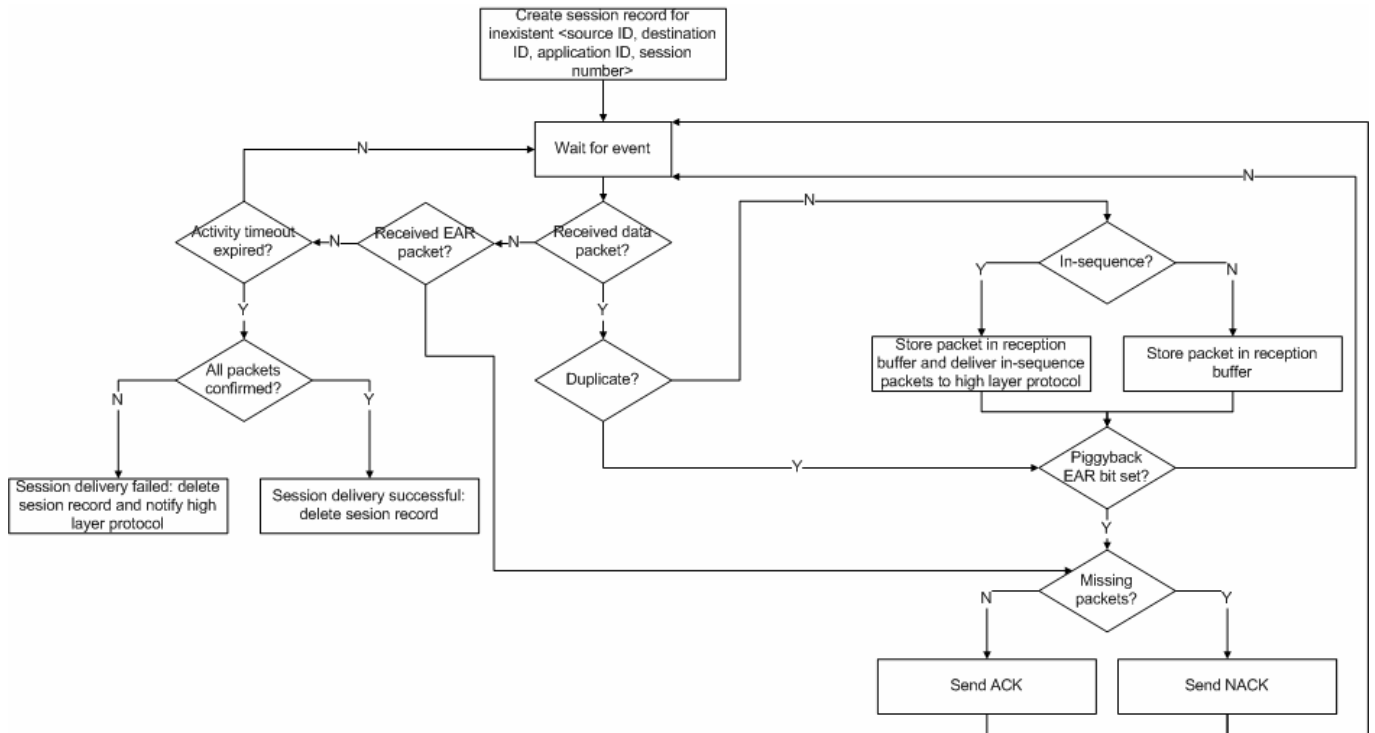


**Figure 1: DTSN total reliability algorithm at the source.**

<sup>1</sup> A new DTSN RTT estimation mechanism is currently under study, which adapts more quickly to the sudden RTT variations, characteristic of wireless multihop networks.

The DTSN algorithm at the destination is depicted in Figure 2. Upon reception of a data packet with a new *session identifier*, a new session record is created. If, on the other hand, the session identifier exists but the *session number* is different from the recorded one, the session record is reset and the new *session number* replaces the old one. The destination then collects the data packets that belong to that flow, delivering in-sequence packets to the higher layer protocol. Upon reception of an EAR, the destination sends an ACK or NACK depending on the existence of gaps in the received data packet stream.

Upon the expiration of an activity timer, the session record is deleted and the higher layer protocol is notified in case there are unconfirmed packets.



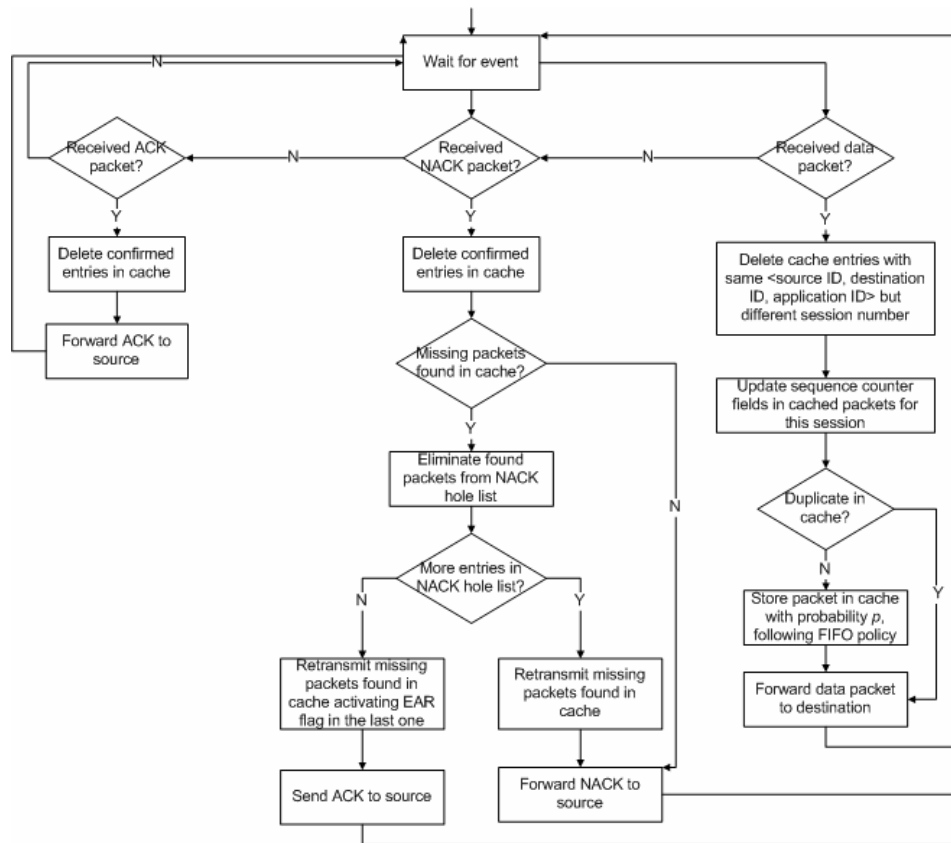
**Figure 2: DTSN total reliability algorithm at the destination.**

As already explained, the strict end-to-end transport reliability model used in TCP is not suited for WSNs because it leads to extra consumption of the scarce bandwidth and energy resources. This is caused by the fact that missing packets (and some control packets like NACKs) are retransmitted end-to-end, expending bandwidth and energy in all links/nodes in the path between source and destination. Caching at intermediate nodes is the mechanism employed by DTSN to counter this inefficiency (see Figure 3). In DTSN, each node keeps a cache of intercepted packets, managed according to a suitable replacement policy, such as FIFO. The packets are stored in cache with probability  $p$ , and may belong to different sessions whose end-to-end routing path includes the node in question. Each time an intermediate node receives a NACK packet, it analyses its body and searches for corresponding data packets that are missing at the destination. In case a missing packet is detected, the intermediate node retransmits it. It also changes the NACK contents before resending it, adapting its gap list so that the retransmitted packets are not included (eventually, the NACK may become an ACK in case all gaps were found in cache). In this way, the source will only have to retransmit those data packets that were not cached at intermediate nodes, decreasing the average hop length of the paths followed by retransmitted packets. Additionally, intermediate nodes eliminate the cache entries that correspond to packets whose delivery is confirmed by an ACK or NACK. Intermediate nodes also

process the *session number* header field in data packets, replacing any cache entries whose *session number* is outdated. In fact, this is currently the mechanism by which event-driven reliability is implemented.

The intermediate nodes may also be configured to keep the confirmation status for each session (not the packets themselves), even after an ACK was received. This reduced set of information allows the node to generate ACKs in response to EARs and to suppress redundant or obsolete data and control messages. For sensors with increased capabilities, the storage of a more detailed set of information could further optimize the DTSN service, e.g., to keep track of the holes in a sequence of packets allows the node to generate NACK packets locally upon reception of an EAR for the session. More advanced functionalities will be studied in the future.

The caching probability  $p$  at the intermediate node should be defined taking into account various factors like cache size, traffic load and EAR frequency. The value of  $p$  must be chosen to maximize the probability of cache hit for the NACK requested data along the path.



**Figure 3: DTSN total reliability algorithm at the intermediate nodes.**

In order to further increase the utilization of the distributed memory capacity of WSN nodes, there is an additional caching mechanism designated *reliable broadcast*. This mechanism takes advantage of MAC layer overhearing so that nodes which are neighbors of those that compose the session's routing path can also perform caching of the data packets, though the packets are transmitted in unicast mode. This technique requires cross-layer collaboration with the MAC protocol. In the case of CSMA/CA based, a flag in the header of RTS and Data frames is used to notify overhearing nodes that they should stay awake, peeking on the data transmission and passing the payload to the transport layer for caching. ACK and NACK are also

transmitted in overhearing mode. When a overheard NACK frame asks for packets stored in the cache of one those neighbors, it transmits the packet towards the destination.

Figure 4 illustrates the transmission of a block of six packets, where the second and fourth packet are lost and then recovered through intermediate caching. It is assumed that  $AW=5$ .

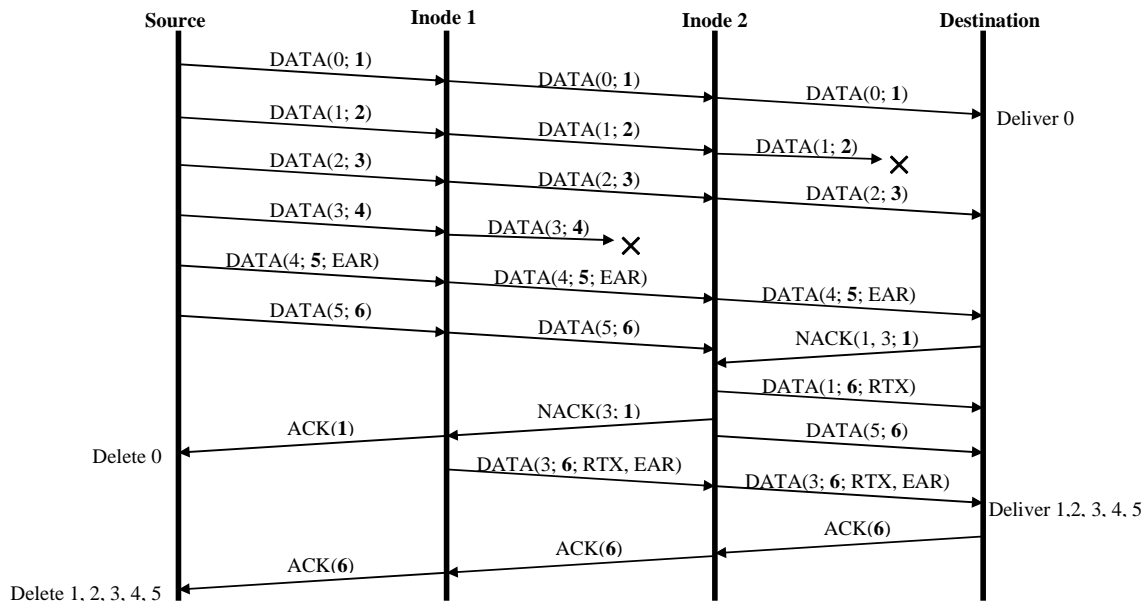


Figure 4: The source transmits a block of 6 packets with  $AW=5$ . Sequence counter values are represented in boldface.

### 3.2 DTSN Differentiated Reliability Service

Total reliability is not always required, which is the case of some kinds of acoustic recording, imaging (where error resilience mechanisms are usually provided at the application Layer) and periodic reporting on non-critical data. Sometimes an increase in energy consumption can be justified by other benefits (high throughput / low latency). In such scenarios, it is possible to modify the basic DTSN ARQ policy, eventually combining it with Forward Error Correction (FEC) strategy.

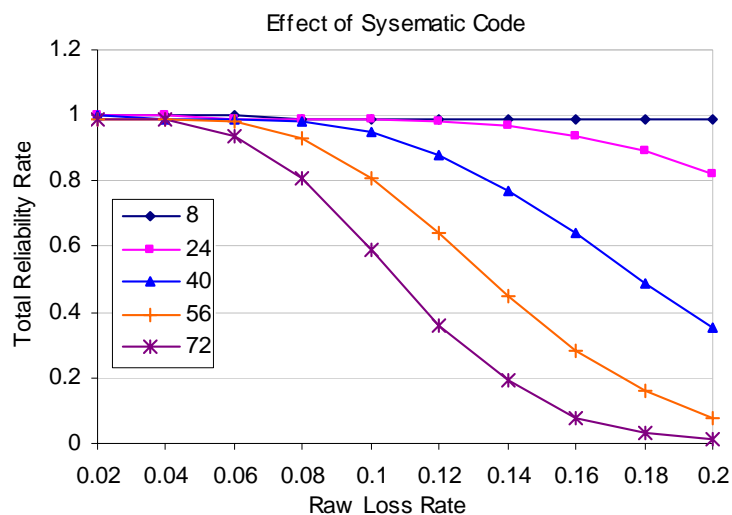
When the data flow can tolerate some losses, the transport layer can exploit this to achieve higher efficiency by adapting its buffering strategy to the specific end-to-end reliability requirements: it can store just a core part of the data, to guarantee a minimum percentage of delivery through the ARQ mechanism, relying on the underlying MAC protocol (which usually supports link reliability) and intermediate caching mechanisms to deliver the remainder with best-effort reliability. This is designated Enhancement Flow strategy. Here, an higher amount of buffering at the source leads to an higher reliability grade. The block of data packets still forms a reliability unit. Within each block, a core of  $i$  out of the total  $n$  packets may be buffered at the source for total reliability. In the case of imaging sensors, this core might be used to guarantee minimum image resolution, while the additional  $n-i$  packets might constitute imaging enhancement layers. In principle, the buffering level at the source can be set even to zero (no core in block) to allow the source to send data continuously without end-to-end loss recovery based on the ARQ mechanism. ARQ exclusively based on intermediate node caching will try to guarantee per se a satisfactory albeit probabilistic reliability grade.

Erasure coding or FEC is another strategy that can be used by DTSN. In a wireless multi-hop network the packet error rate (PER) over a single link is influenced by several factors, such as the physical error rate, the sleeping period of the nodes (transceiver off) and, especially, co-channel interference. The resulting PER is expected to fall within the interval of 5% and 10% [9]. Considering a transmission over  $n$  hops, the probability of success  $P_R(n)$  is:

$$P_R(n) = (1 - PER)^n, \quad (1)$$

In [9], the authors have shown that the expected end-to-end packet delivery rate decreases very sharply with the number of hops, for a link PER higher than 1%. This motivates the use of FEC besides intermediate caching to increase loss tolerance [10]. It can be shown that when the transport layer operates in the region characterized by a raw packet error rate in the interval [0%; 10%], the addition of a few parity packets contributes significantly to decrease the final packet loss ratio. Consequently, the use of a FEC code has a real impact on reliability, while introducing a limited overhead.

Figure 5 shows the probability of successful recovery of a whole data block when  $k$  code words are received out of  $n$  originally sent. The overhead associated to the FEC coding can be reduced up to 10% ([80, 72] code) but its efficiency decreases quickly with the loss probability.



**Figure 5: Impact of a FEC code, which adds 8 redundant code words to [8; 72] original messages.**

It should be emphasized that the FEC option can be used together with the Enhancement Flow option, since besides increasing reliability, it can improve throughput by reducing the time needed for reconstructing data at the sink (though this is somewhat compensated by the FEC overhead).

However, simulation studies have revealed that the effectiveness of these techniques is maximized when transferring large data blocks under low loss rates (to reduce the overhead of parity packets and to achieve high grades of partial reliability), no congestion (when the network is congested, throughput must not be increased) and high end-to-end latency (to justify the loss of total reliability with a consistent improvement of the throughput).

## 4 Simulation Results

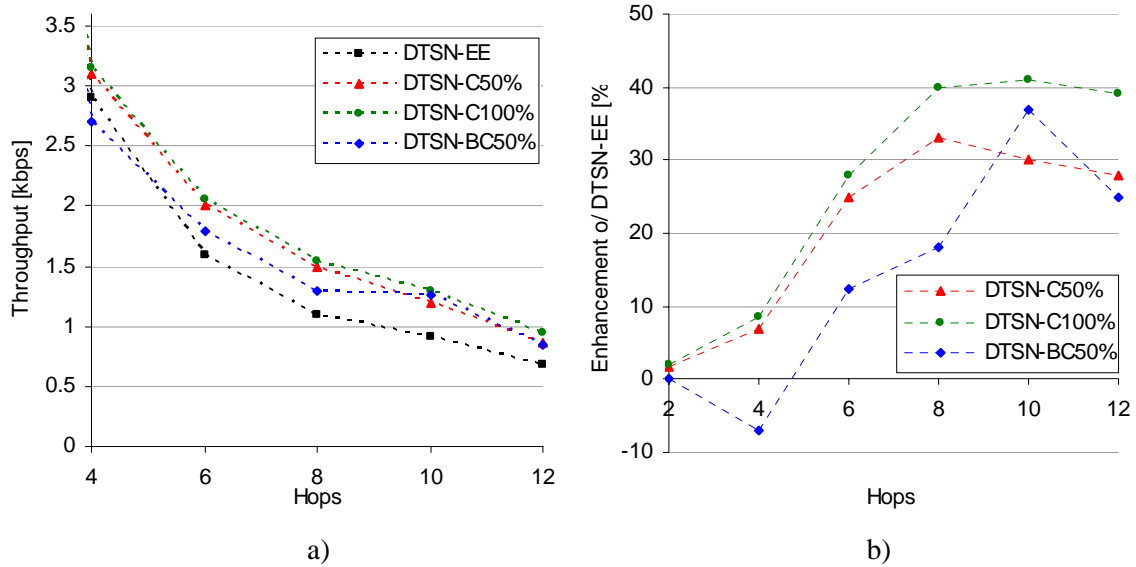
DTSN was simulated in OMNET++ [11], a discrete event component-based simulator that is quickly gaining interest in WSN simulation community. OMNET++ allowed the integration of DTSN with the MAC Simulator, developed by the T.U. Delft, which provides implementations of several MAC layers protocols (CSMA/CA, T-MAC, S-MAC etc.) and a simple omniscient routing algorithm. The implemented DTSN module includes almost all the functionalities of DTSN.

### 4.1 Simulation 1: DTSN Total Reliability Service

The aim of this simulation is to estimate the advantages on caching at intermediate nodes regarding throughput and energy-efficiency. The scenario considers a single source sending 10000 octets of data (corresponding to a sequence of 100 data packets of 100 octets each, the size of the output buffer) toward a sink located several hops away. Data is transferred with the total reliability service. The PER in each hop is fixed at 10%. The DTSN runs in conjunction with the CSMA/CA MAC protocol, configured to have the maximum number of transmission attempts set to 4. The AW size is set at 10000 octets. The intermediate node cache size is equal to 5000 octets (50 data packets).

The throughput results are depicted in Figure 6, where a purely end-to-end DTSN configuration is compared with three intermediate caching configurations: caching at intermediate nodes in the path with  $p=50\%$  (DTSN-C50%); caching at intermediate nodes in the path with  $p=100\%$  (DTSN-C100%); and caching that alternates between *reliable broadcast* and caching at intermediate nodes in the path, with 50% of probability each (DTSN-BC50%). The results clearly show that DTSN-C50% already introduces a significant enhancement (25%) when the source and destination have a distance of more than 6 hops, achieving a peak at 8 hops. With DTSN-C100%, the enhancement can be as high as 40% at 10 hops. In both configurations, the increase in performance stabilizes or even decreases slightly after the peaks. We believe that the limit is imposed by data overwriting rate at intermediate caches, since the caching buffer is one half of the AW. This causes a significant number of end-to-end retransmissions.

Regarding DTSN-BC50%, it was not as effective as initially expected. In this case, the NACK packets triggered several path neighbors to contend to send the requested packets that were in cache, negatively compensating the benefits of extra caching.



**Figure 6: Results for simulation 1. a) Throughput; b) Enhancement of intermediate caching over DTSN-EE.**

Regarding energy-consumption, the costs were calculated in accordance with the values for the MICAz module, which are reported in Table 1.

**Table 1: Energy consumption parameters for the MICAz mote devices.**

<b>Current Draw</b>	Receive mode	19.7 mA
	Transmission mode (0 dBm)	17.4 mA
	Idle mode	20 uA
<b>External Power</b>		3.3 V

As shown in Figure 7, intermediate caching achieves the goal of reducing end-to-end transmissions to some extent, and consequently the energy consumption. Energy savings become more consistent as the distance to the sink increase. However, the improvements are not as relevant as expected due to overhearing and hidden terminal problems of CSMA/CA, which significantly compensate the effect of intermediate caching (the adaptation of more efficient MAC protocols is envisaged for future work). The two versions of DTSN did not differ considerably: the version with *reliable broadcasting*, even introducing extra costs due to contention, allows the recover of a great number of segments from intermediate caches, while DTSN-C100%, even not introducing those broadcasting extra costs, demands longer retransmission paths because some messages are often overwritten in intermediate caches before being requested by NACKs.

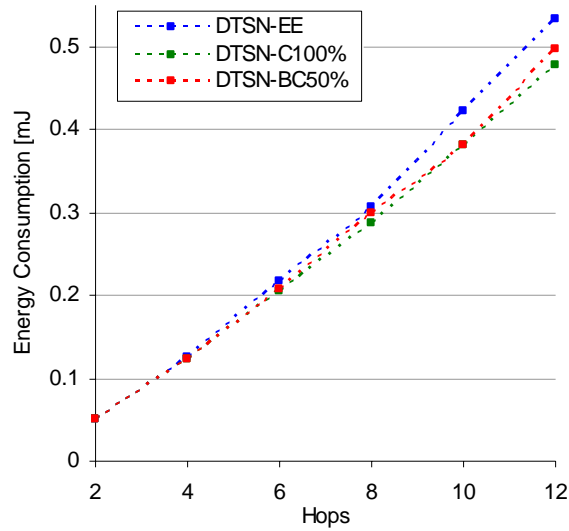


Figure 7: Energy consumption results for simulation 1.

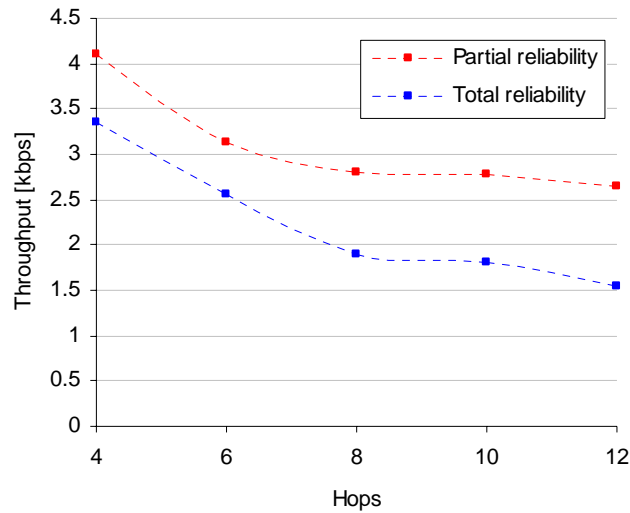
## 4.2 Simulation 2: Enhancement Flow and FEC

The impact of the Enhancement Flow and FEC coding options on DTSN performance was also evaluated in terms of throughput and data block integrity at the destination, as a function of the number of hops between source and destination. For the partial reliability service, data blocks had 50% of core packets and 50% of enhancement packets. Each block was also encoded with a Reed-Solomon code of rate  $r = 5\%$ <sup>2</sup>. For this configuration, the intermediate caching probability was  $p = 50\%$ . The block size is fixed at 50000 octets. All the other parameters were the same as used in the first simulation.

Figure 8 depicts the comparison of partial reliability and total reliability in terms of throughput. The values of the throughput only consider the useful data segments received at destination. The advantages of the Enhanced Flow option are clear. With total reliability all the transmitted segments are buffered at the source until a confirmation of reception is received. When the output buffer is full, the source stops sending and waits a confirmation to avoid buffer overflow. As the RTT increases, these inactivity periods become longer and more frequent, significantly affecting the throughput. Enabling the enhancement flow option, it is less probable that the source fills the output buffer before the reception of a confirmation, so, it can push data in a continuous way, achieving higher values of throughput.

---

<sup>2</sup> When using a FEC block code, the  $k$  original data words/packets are encoded into a total of  $n = l + k$  code words/packets, where  $l$  is the number of redundant words/packets. The code rate  $r$  is calculated as  $r = \frac{l}{k}$ .



**Figure 8: Throughput comparison between the total reliability and partial reliability DTSN services.**

Reliability of communications is another important parameter to estimate the performance of DTSN. The delivery of the enhancement data is not guaranteed but, combining intermediate caching with FEC, a high level of reliability is provided. Simulations have shown that the probability of success for the transfer of data blocks of 10 kBytes (50% of core packets and 50% of enhancement packets) was around 90% independently of the number of hops. This means that even if the effectiveness of the FEC decreases with the distance (the probability of losing a packet increase with the number of hops and the FEC has a limited correction capability), the number of nodes involved in the intermediate caching increases so the probability of finding missing segments along the path also increases

The interaction between the intermediate caching and the FEC was also evaluated looking at the integrity level of the received data. The results revealed that for those blocks that were correctly decoded, the number of received packets was above 95% of the original.

## 5 Conclusions

This paper presents DTSN, a new reliable transport protocol for WSNs. DTSN supports different grades of reliability. Total reliability is based on end-to-end Selective Repeat ARQ, coupled with caching of data packets at intermediate nodes so that the number of end-to-end retransmissions is minimized. Partial reliability can be achieved by the Enhancement Flow and FEC options. The Enhancement Flow option consists of buffering only a fraction of a block of data packets at the source (designated the core), being transmitted with total reliability. The remaining data packets that constitute the block are granted no guarantees, since they are considered enhancement data. When coupled with intermediate caching and/or FEC, the Enhancement Flow is able to achieve high reliability grades while significantly increasing the throughput in comparison with the total reliability service.

An implementation of the DTSN total reliability service for TinyOS was already developed within the framework of the IST UbiSec&Sens project and is currently under evaluation.

As future work, the authors plan to compare DTSN with other WSN transport protocols like nano-TCP, RMST, DTC and ESRT. The authors also plan to improve DTSN, including the addition of a congestion control mechanism.

## Acknowledgements

The work described in this paper is based on results of IST FP6 project UbiSec&Sens. UbiSec&Sens receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper.

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

## References

- [1] Postel, J. Transmission Control Protocol. RFC 793, IETF, September 1981.
- [2] Wang, C.; Sohraby, K.; Li, B.; Daneshmand, M.; and Hu, Y. *A Survey of Transport Protocols for Wireless Sensor Networks*. IEEE Network, May/June 2006.
- [3] Stann, F.; and Heidemann, J. RMST: Reliable Data Transport in Sensor Networks. 1<sup>st</sup> IEEE International Workshop on Sensor Net Protocols and Applications, Anchorage, Alaska, USA, May 2003, pp. 1-11.
- [4] Intanagonwiwat, C.; Govindan, R.; and Estrin, D. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCON), August 2000.
- [5] Sankarasubramaniam, Y.; Akan, O.B.; and Akyildiz, I.F. ESRT: Event-to-sink Reliable Transport in Wireless Sensor Networks. Proceedings of the ACM MobiHoc '03, Annapolis, Maryland, USA, June 2003, pp. 177-188.
- [6] Dunkels, A.; Alonso, J.; Voigt, T.; and Ritter, H. Distributed TCP Caching for Wireless Sensor Networks. Proceedings of the 3<sup>rd</sup> Annual Mediterranean Ad-Hoc Networks Workshop, Bodrum, Turkey, June 2004.
- [7] Tezcan, N.; and Wang, W. *ART: An Asymmetric and Reliable Transport for Wireless Sensor Networks*. Accepted for publication in Journal of Wireless Sensor Networks (JSNET), April 2006.
- [8] Jacobson, V. Congestion avoidance and control. Proceedings of the ACM SIGCOMM'88, Stanford, CA, USA, August 1988.
- [9] Wan, C.-Y., Campbell, A., Krishnamurthy, L. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. Proceedings of WSNA'2002. Atlanta, Georgia, USA, 2002.
- [10] Kim, S.; Fonseca, R.; and Culler, D. Reliable Transfer on Wireless Sensor Networks. Proceedings of the 1<sup>st</sup> Annual IEEE Communications Society Conference on Sensor Ad-Hoc Communications and Networks (IEEE SECON 2004), Santa Clara, CA, USA, October 2004, pp. 449-459.
- [11] Varga, A. The OMNET++ Discrete Event Simulation System. Proceedings of the European Simulation Multi-conference (ESM'2001). Prague, Czech Republic, 2001.