Multimedia Data Transport for Wireless Sensor Networks

João Almeida, António Grilo, and Paulo Rogério Pereira

Abstract—**This paper presents the study and implementation of a reliable transport protocol for wireless multimedia sensor networks. The work was based on an existing transport protocol for sensor networks, the DTSN ("Distributed Transport for Sensor Networks") developed at Inesc within projects IST FP6 UbiSec&Sens, EuroFGI and FP7 EuroNF.**

For multimedia data transport, some structural changes were made to DTSN, and mechanisms were added in order to provide a differentiated reliability service for packets flowing on the sensor network. The new service allows a multimedia data stream to be divided on several blocks with different reliability. The blocks' size and minimum reliability required are sent to the network by the client to configure and initialize the differentiated reliability service. Lost packets from multimedia blocks with the minimum reliability already assured will not be claimed lost by the receiver, and therefore will not be retransmitted. That will cause fewer retransmissions on the network, throughput improvements and an increase of network lifetime with no significant degradation of the transmitted stream.

A packet recovery module was also implemented, based on an additional parity packet that must be sent for each block, allowing the recovery of one lost packet per multimedia block. This module works on top of the transport layer, improving overall efficiency and being completely transparent to it.

The new multimedia DTSN protocol was tested in the TOSSIM network simulator. Functional and performance tests were made which have demonstrated the advantages of the proposed solution for multimedia transport in wireless sensor networks.

Index Terms—**Wireless Sensor Networks, Transport Protocol, Multimedia, Differentiated Reliability, Packet Recovery.**

I. Introduction

DURING the last few years, interest in Wireless Sensor Networks (WSN) [1] has grown significantly. The component prices getting cheaper, as well as other external factors, amongst which the increased demand for public security (like improved video-surveillance) and the need for improved efficiency of health care services have motivated the research efforts in this area.

However, the limited resources of this kind of network nodes are one hard barrier that developers have to face. In order to build protocols that allow WSNs to carry average amounts of multimedia traffic [2], developers must take into account the significant packet loss ratio inherent to low-power radio-frequency communications. While the communication protocols provide mechanisms for the retransmission of lost data packets, this significantly increases the transmission time (decreasing throughput levels), as well as decreasing the network lifetime. One strategy to circumvent these obstacles is to partition multimedia streams into layers or blocks with different degrees of importance regarding the final quality of service perceived by the user, associating them with different degrees of reliability. In this way, the network protocols are able to know which traffic is critical and which is non-critical.

Based on these concepts, a solution is proposed in this paper – a transport protocol that according to the traffic importance – which is defined by the application layer – presents different behaviors regarding lost packets.

The presented work is focused on the DTSN [3] (Distributed Transport for Sensor Networks) transport protocol, though most of the developed techniques could easily be adapted to other WSN transport protocols. While the DTSN basic guaranteed delivery service was already specified and implemented, the differentiated reliability service was little but sketched. The development of this differentiated reliability service became the focus of the work presented in this paper. Once this service was implemented, new ideas raised, namely to complement this differentiated service with a packet recovery mechanism based on forward error correction (FEC) coding [4]. With that mechanism implemented, packets lost and not retransmitted (belonging to a non-critical block) could be recovered with the help of a specific packet containing the FEC code.

The rest of this paper is organized as follows. In section II, a short presentation of DTSN and its main characteristics is provided. Section III focuses on the description of DTSN's new differentiated reliability service. Section IV analyses the packet recovery techniques implemented, with main focus on the changes made to the protocol structure and the way coding/decoding was accomplished. Section V presents simulation results. Finally, some conclusions will be drawn, commenting on the achieved results and raising suggestions for future work.

II. DTSN

DTSN's main goal is to guarantee a reliable end-to-end communication on Wireless Sensor Networks. The well

Manuscript received January 25, 2009, revised April 13, 2009.

J. Almeida, A. Grilo and P. Pereira are with INESC-ID, Instituto Superior Técnico, Technical University of Lisbon, Rua Alves Redol, nº 9, Lisboa, Portugal (phone: +351-213100226; fax: +351-213145843; e-mails: joaocalmeida@ist.utl.pt, antonio.grilo@inesc-id.pt, prbp@inesc.pt).

known resource limitations of those networks (limited power supplies, tiny mote memory, etc…), were important factors in the design of this protocol. High levels of scalability and robustness were also important goals of this protocol.

DTSN assumes that the MAC-layer provides a reliable link and that the Network Layer supports symmetric bi-directional paths (i.e. acknowledgement packets flow back to the source through the very same nodes that forward the data packets)[1].

A. DTSN basic service

DTSN full reliability service provides 100% guaranteed delivery of all packets to the destination.

Like in TCP, the DTSN transport protocol is based on the concept of end-to-end sessions. However, the sessions are soft-state. A session is started whenever a node sends the first packet to a new destination and is closed whenever an activity timer expires. A session is defined by its session identifier which consists of the source address, destination address, application/service identifier (similar to the port number in TCP) and session number (session sequence number, which allows the endpoints to identify when a session is started anew).

B. Sender

In DTSN, packets are sent until the sender reaches the limit of its transmission window (known as DTSN window). The size of this window may vary with many factors, ranging from the network topology to the type of external environment. Another window – the acknowledgment window – is used to determine the number of packets sent before sending an EAR ("Explicit Acknowledgment Request") packet. EAR packets may originate two kinds of replies from the destination: NACK or ACK. While ACK confirms the successful reception of all packets till the end of the last acknowledgment window, receiving NACK packets warns the sender that some packets have been lost. The sequence numbers of lost packets are marked on NACK packets. Once an EAR packet is sent, an EAR timer will be launched. If the timer expires before a reply is received from the destination node, the sender will repeat the EAR transmission until a defined maximum number of EAR retries have been reached. If this number is reached, the sender will close the session and inform the upper layer that some packets were unconfirmed.

Sent packets are kept on a session specific buffer until its reception is confirmed with an ACK packet. If a NACK packet arrives, the sender will then identify the packets that have been lost and immediately retransmit them (selective repeat).

C. Receiver

In the DTSN basic service, the receiver creates a new session whenever it receives traffic with a new session identifier. After receiving packets from the lower layer, those will be stored in a reception buffer and ordered by sequence number. If the sequence number received is the one expected, the packet is delivered to the upper layer, otherwise it will remain buffered until the missing packets arrive. This behavior will allow packets to be delivered to the application layer in sequence.

When an EAR packet arrives, the receiver will look for gaps in buffered packets. If there is any gap, it will reply with a NACK packet containing the missing sequence numbers. Otherwise, it will send an ACK packet confirming that all packets have been successfully received.

If the session's activity timeout expires, the session will be removed and the upper layer informed of missing packets, if any.

D. Intermediate Nodes

Depending on the network topology, the transmission power, the receiver's sensibility, or external environment issues, network traffic may have to flow across several intermediate nodes before arriving to the destination. In DTSN, the role of intermediate nodes is leveraged in order to reduce the number of retransmissions and the effect of lost packets on the network throughput. In order to achieve that, intermediate nodes cache a sample of packets that flow through them. Whenever a NACK packet crosses an intermediate node, the cache is checked for the destination's missing packets. The intermediate node will transmit the cached missing packets to the destination, removing them from the missing packets list in the NACK. If necessary, the NACK may be changed to an ACK packet which will flow all the way to the sender. That behavior will avoid end-to-end retransmissions, transforming them in local retransmissions, saving time and energy.

Cache management is very important. Since the cache dimension is limited, it is essential to keep the list of cached packets updated, saving the maximum of unacknowledged destination packets and trashing the obsolete ones. Whenever an ACK packet is received, confirmed packets are deleted from the cache. Packets with lower session numbers are also removed from the cache if an incoming packet with the same origin / destination address pair but with a higher session number arrives from the sender.

In the future, these operations may be combined with distributed sampling of unconfirmed packets in intermediate nodes in order to increase the probability of cache hits and avoid even more end-to-end retransmissions.

III. DTSN'S DIFFERENTIATED RELIABILITY SERVICE

Following what was explained above, although the DTSN protocol was carefully built in what concerns WSNs limitations, the basic service (i.e. the total reliability service) is not suitable at all for multimedia traffic.

The multimedia extension divides each media frame at the sender node, into several blocks, each one with a different reliability level. The reliability level is here defined as the minimum fraction of packets of a block that must be correctly delivered to the destination so that the corresponding multimedia frame achieves the minimum acceptable quality

[1] In fact, DTSN can operate on top of a routing layer that supports asymmetric paths, since it follows the conventional end-to-end philosophy for guaranteed delivery. However, symmetric paths improve its performance allowing it to take advantage of intermediate node caching.

when rendered.

The reliability levels shall be configured by the application in order to, depending on the multimedia traffic type to be transmitted (e.g. encoding format), assure that blocks that carry critical data (e.g. encoding / decoding information) arrive complete to the destination, while other less important blocks (e.g. those containing pixel values) may only partially arrive.

A. Message format

The multimedia extension packets carried in basic DTSN payloads can be either session configuration or data packets. The message structure is shown in Fig. 1.

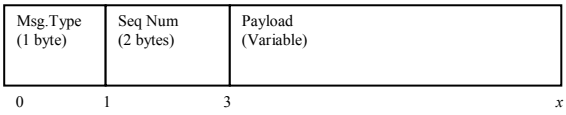

Fig 1. Application layer message structure.

The message type field identifies the type of packet sent. Type 0 is for data packets, while type 1 is for configuration packets. The Sequence Number is extremely important for the differentiated reliability service. This issue will be discussed below.

The payload of configuration messages indicates both the sizes and reliability levels of the blocks that belong to the frame. This message structure is presented in Fig. 2.

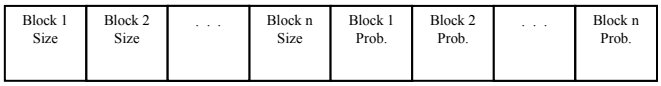

Fig 2. Payload structure for an application layer configuration packet.

B. Protocol behavior: Sender

For each frame – which corresponds to an independent DTSN session –, the number, sizes and reliability levels of the blocks are provided by the sender application to the DTSN layer using a special S*essionConfiguration_REQ* primitive, invoked before actually sending the data packets. When the *SessionConfiguration_REQ* primitive is invoked from the application layer, the DTSN sender creates a new session (increasing the session's number in the session identifier). Correspondingly, it builds a session configuration packet (see above), sending it to the receiver. All the data packets that are received from the application layer for that session are then sent normally as in the basic service.

C. Protocol behavior: Receiver

When a destination node receives a packet, it checks whether it is configuration or data. In the first case, the receiver (re-)starts the session for the received sender address, receiver address and application/service identifier, creating a list of block elements. This list contains the size, reliability level and number of packets actually received for each block. Then, the DTSN receiver notifies the application layer about session configuration, using the *SessionConfiguration_IND* primitive.

As data packets start to arrive at the destination node, the receiver starts to increment the number of packets (on the block elements list) accordingly. Packets can be received out of order. The block to which each packet belongs to is known by comparing the received packet's sequence number with the range of sequence numbers of each block. The number of packets in that block is increased. If the number of packets received for that block has reached or exceeded the minimum reliability for that block (defined in the session configuration, as mentioned above), the block is signaled as having attained the minimum reliability level by setting the special value (0xFFFF) to the number of packets received for that block. In this way, the block will be handled properly upon arrival of the next EAR.

When an EAR packet is received, the receiver checks if the EAR sequence number is the one expected. If so, the receiver sends an ACK to the sender. Otherwise, a check will be made on the number of packets of the current block. If that value is not set to the special value that signals that the minimum reliability level has been achieved (0xFFFF), the node sends a NACK back to the sender with a list of all missing packets. Otherwise (i.e. if the number of block packets is set to 0xFFFF), the node will assume that block is complete, responding with an ACK packet. In this case, the expected sequence number is set to the first packet of the next block. The reception of the ACK will make the sender act as if all packets sent earlier were received successfully by the destination, preventing the eventual retransmission of still missing packets belonging to that block. However, the number of packets received at the destination node may not be equal to the block size. In fact, packets received were just in sufficient number to make the receiver consider that the minimum defined reliability was reached.

Whenever a block reaches its minimum reliability level, data packet delivery to the application resumes even if there are holes in the packet sequence. The receiver application analyses the sequence number of each packet to see if it is the one expected. If not, that means that there is at least one missing packet that belongs to a block whose minimum reliability was less than 100%. The importance of the sequence number of Fig.1 is obvious. Without it, the application would not know the place of the packets in the frame, since the differentiated reliability service implies that the packet sequence may not be continuous as in the basic service.

Fig. 3 shows an example of DTSN differentiated service behavior. The application configures blocks of size 5 packets with 80% reliability. Packets 3 and 4 are lost, which is below the minimum reliability. So, the EAR packet triggers a NAK with the missing packets list. Even though packet 4 is lost again, a new EAR will find 4 out of 5 packets present, so they are delivered to the application layer, saving subsequent retransmissions of packet 4.

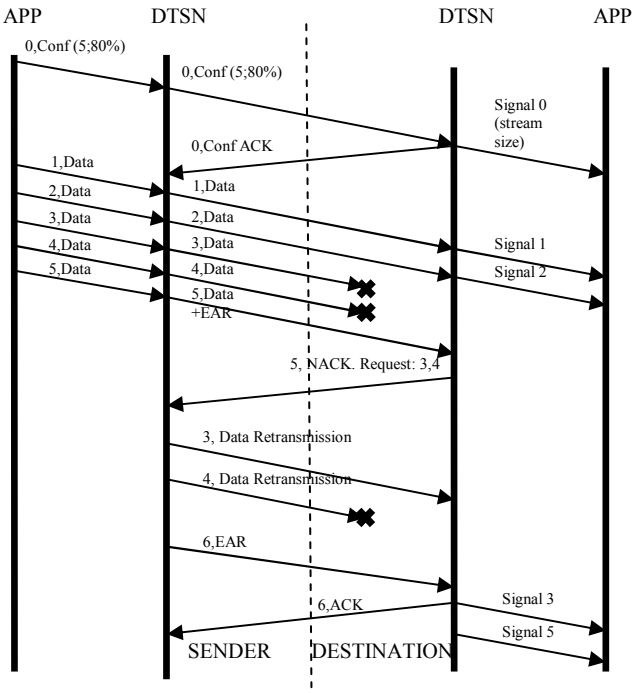

Fig 3. Example of DTSN differentiated service behavior.

IV. PACKET RECOVERY MECHANISM – FEC

DTSN's differentiated reliability service made the transport of multimedia traffic along wireless sensor networks feasible.

As explained before, missing packets belonging to blocks which already have reached minimum reliability will not be retransmitted.

If somehow, packets lost could be recovered without interfering with DTSN normal behavior, that would increase the system throughput by not having the network flooded with retransmission and control packets. A way of achieving this goal is the implementation of a packet recovery layer using forward error correction (FEC) mechanisms, placed between the application and transport layers. This FEC layer must be transparent to both those layers (application and transport) to make sure it will not interfere with them.

This solution is based on the encoding of a block with FEC, using an erasure code. Erasure codes are FEC codes specially designed to deal with lost packets (as corrupt packets will be discarded). As a proof-of-concept, a simple parity packet was implemented as the erasure code.

A. Erasure code

The packet recovery layer creates a parity packet per multimedia block, performing a byte by byte XOR operation with all packets in the block. Hence, this parity packet allows the recovery of one lost packet per block. Although this FEC code may appear to have weak packet recovery capabilities, they depend on the multimedia block size. The shorter the blocks are, the stronger the packet recovery capabilities are. The price is the redundancy added which becomes more significant.

Additionally, the FEC layer is coded and inserted on the layer structure in such a way that it is easy to replace the current erasure code by a more powerful one, in order to allow recovering a larger number of packets.

B. Transparency

As said before, the FEC layer is placed between the transport and application layers. Messages exchanged between these two layers now pass by this new layer. The main focus of the transparency issue was making sure that the transport layer does not have to know about FEC layer operations.

Another issue that now comes up is the following: if the erasure code only allows the recovery of one lost packet, what needs to be done for blocks whose minimum reliability allows more than one packet to be lost?

The first step was programming the FEC module so it would be able to check the reliability for each block in the network configuration packet. For blocks in which the minimum reliability is configured to allow more than one lost packet, the FEC layer on the sender will not generate the parity packet. In this case, the FEC receiver will also not try to recover any lost packets. Therefore, for this case, FEC packet recovery will simply not be performed.

On the other hand, if the block's configured reliability allows just one or no losses, the configuration packet will be modified in order to configure DTSN differentiated reliability service to allow one lost packet in that block. In this way, DTSN will work as before, based on the configuration values modified by the FEC layer and the FEC layer will recover the missing packet if necessary.

In this way, the transparency goal is achieved. The existence of the FEC layer is not noticed by either the application or the transport layer.

C. Packet encoder – Sender

The FEC layer on the sender side is responsible for performing a byte by byte XOR operation on each block message coming from the upper layer (application). The result of this operation is sent as the payload of the last packet in each block.

D. Packet decoder –Receiver

At the receiver, the FEC layer is responsible for receiving packets from DTSN (before sending them to the application) and for making a byte by byte XOR operation on their payloads. The FEC layer will check for gaps in the received sequence numbers. If the sequence number received was not the one expected, surely some packet is missing. In this case, a recovery procedure will take place until only one packet is missing in the block. If only one packet is missing, the result of the XOR operation will be the payload of the missing packet which is promptly signaled to the application layer, followed by the remaining packets in the block. The parity packets (last packet of each block) are not signaled to the application layer.

Fig. 4 shows an example of the packet recovery system behavior. In this example, the application generates a 4 packet data block with 100% reliability. The FEC layer configures a 5 packet data block with 80% reliability, where the fifth packet

in the block contains the FEC data. Two packets are lost, triggering a NACK. When one of the missing packets is recovered, the DTSN layer achieves the configured 80% reliability and delivers the packets to the FEC layer. The FEC layer recovers the missing packet (packet 4) and handles them to the application layer.

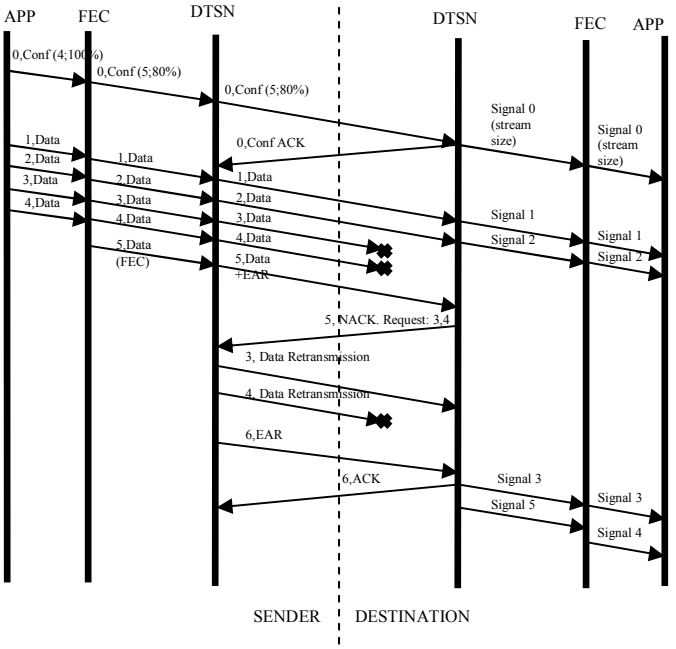

Fig 4. Example of FEC layer behavior.

E. *Cross layer Interactions*

For the packet recovery procedure, the FEC layer uses the packets stored on the transport layer receiver buffer in order to recover the missing packet. This is a very good example of how cross layer interactions are important in software development for wireless sensor networks. Without this interaction, memory would not be efficiently used as packets would also have to be stored in the FEC layer. Moreover, transparency would be at risk – In order to store all block packets in the FEC layer, DTSN would have to signal them all whether they were ordered (ready to send to application layer) or unordered (remaining at the buffer). That procedure would of course change DTSN protocol behavior which is against the transparency goal.

V. Simulation Results

In order to test the developed DTSN differentiated reliability service efficiency on improving system throughput by reducing delays due to retransmissions, a simulation environment was created.

A. *System throughput simulation*

For system throughput simulation, TOSSIM [5] – a TinyOS simulator for wireless sensor networks – was used. A linear network topology of evenly spaced sensor nodes with a variable number of nodes was considered. A transmission bitrate of 250 kbps as in the IEEE 802.15.4 standard for the MAC and physical layers was used [6]. The path loss between each pair of nodes is 72 dB (this corresponds to approximately 10 meters of distance assuming a log-distance path loss model with a distance exponent of 3) and the background noise level is given by the "Meyer Heavy" noise samples provided by TOSSIM.

One hundred packets of one hundred bytes each were divided into five multimedia blocks, all with the same reliability level. System throughput for different numbers of hops and different reliability levels is shown on Fig. 5. Throughput is defined as the ratio of delivered information per unit time.

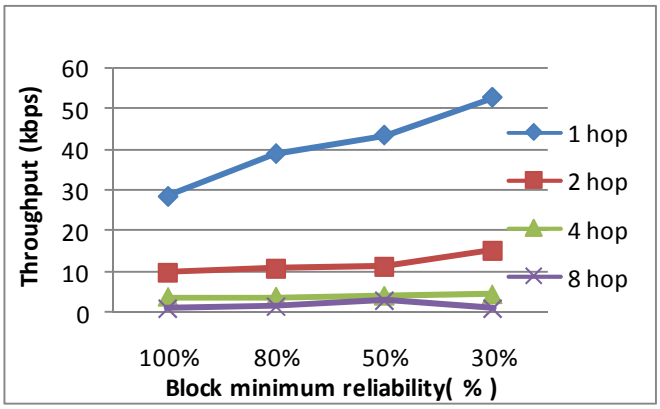

Fig 5. Throughput variation for a full reliable stream and a partially reliable stream.

As was expected, in most cases, throughput increases when the minimum reliability is decreased. As a matter of fact, if the reliability is decreased, less packets have to be retransmitted, resulting in a faster delivery of the packets that need to be delivered.

However, for 8 hops, throughput has decreased when the minimum reliability varied from 50% to 30%. A probable explanation for that behavior is the fact that for 8 hops, the time required for the first-attempt end-to-end transmission of the five blocks is very long compared with local packet retransmissions (i.e. retransmissions obtained from the caches of intermediate nodes in the vicinity of the destination), based on which the 50% reliability level leads to the successful delivery of more packets than with the 30% reliability level. Those factors also explain that the slope in Fig. 5 decreases when the hop distance between the message sender and the destination increases.

Fig. 5 presents the performance results for simulations in a five block multimedia stream, all with the same reliability. Although this test is useful to show how reliability values influences transmission, in the real world it is more probable that an application will require mixed block reliability. Instead of configuring DTSN differentiated reliability service for a stream with the same reliability for all blocks, it will set different values of reliability for each block of the stream. Hence, a second simulation test was performed with 100 packets stream (again five blocks with 20 packets each one), with 100 bytes each packet, but now with a reliability

configuration of 50% for the first, third and fifth blocks (simulating less important packets) and 100% for the second and forth blocks (high importance packets). The results are presented in Fig. 6.

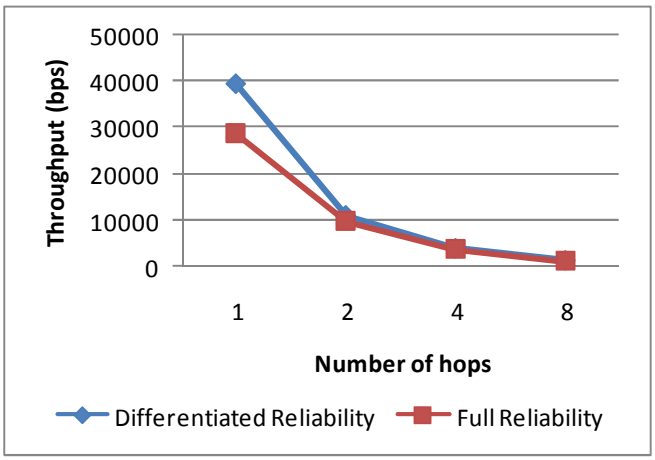

Fig 6. Throughput variation with the number of hops for full reliability and differentiated reliability.

As we can see, the configured partially reliable stream was transmitted at a higher rate than the full reliable stream. However, and as it had already happened in the previous simulation, the difference in throughput is smaller as the number of hops increases.

If we look at the transmission time, shown in Fig. 7, we observe that the transmission time is larger when the full reliability is used, especially for a higher number of hops. Although the throughput is about the same in both cases for 8 hops, transmission time is saved when only partial reliability is required.

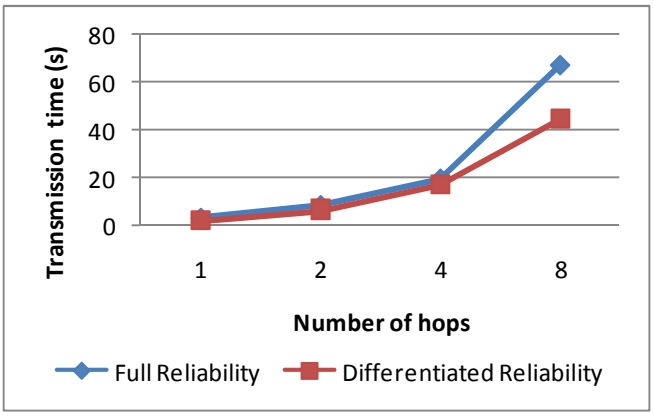

Fig 7. Transmission time variation with the number of hops for full reliability and differentiated reliability.

Now that the differentiated reliability service was successfully tested, it is time to check the performance of the developed packet recovery layer.

In order to leverage FEC layer recovery mechanisms, the network has been configured for a test stream with 5 blocks of 5 packets one hundred bytes each. The reason why the number of block packets was reduced for this test is simple: as our erasure code generates a parity packet that can only recover one lost packet, the shorter a block is, the higher is the probability that only one packet is lost.

As expected in theory, although only slightly, the packet recovery mechanism pumps up throughput, as shown in Fig. 8. Again, throughput decreases with increasing hop distance. As the number of hops increases beyond 3 hops, the packet loss probability increases, resulting in an increase of the probability of more than one packet being lost per block, so that the FEC mechanism is no longer efficient. To improve efficiency, the blocks could be reconstructed at intermediate nodes. This was left for future work.

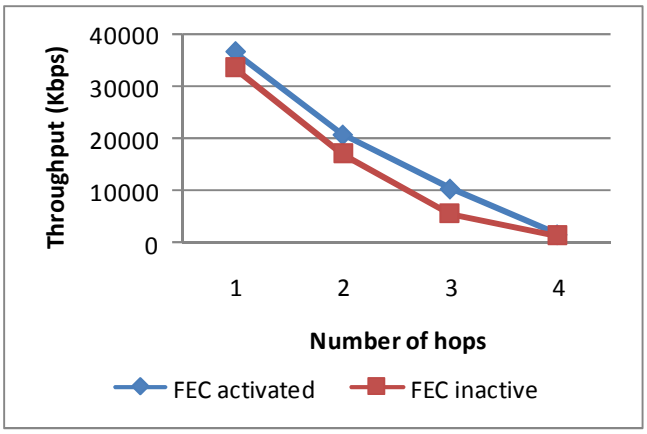

Fig 8. Throughput comparison between a packet recover enabled and disabled systems.

B. *Minimum Reliability vs Real Reliability*

Another important test is to compare the Real Reliability achieved in the simulation with the minimum reliability that was configured for each block. Results are presented in table 1.

MINIMUM RELIABILITY	REAL RELIABILITY
100%	100%
80%	90.5%
50%	75.25%
30%	49.0%

Table 1. Comparison between real and minimum reliabilities.

As can be seen, the real reliability is always greater than the minimum requested reliability (except for 100%, obviously). These results prove that configuring a minimum reliability for one block is not limiting the number of packets it will receive. In fact, and as explained before, the protocol will accept all the block packets even if the minimum reliability is already reached. It is important to remind that reliability levels are just set to avoid retransmissions of unimportant packets.

C. Memory usage

Limited memory capacity is a well known restriction in wireless sensor networks.

DTSN's differentiated service and packet recovery mechanism improve, as discussed before, network throughput and reduce traffic, avoiding congestions. However, the implementation of this system requires additional memory usage. As dynamic memory allocation is not affordable in TinyOS, a RAM usage balance will be made taking into account the memory pools (amounts of memory reserved for a certain variable/structure) created to support DTSN's new differentiated reliability service and the FEC recovery mechanism.

As explained in earlier sections, DTSN differentiated reliability service uses block lists to keep multimedia block information. The FEC layer also uses a similar list in the sender side, in order to keep track of blocks size and when to send the parity packet to the destination. Those two lists are the only structures which needed pools. However, another type of reserved memory cannot be forgotten: the registers used to code / decode parity packets in the FEC layer.

Equation 1 shows the way RAM usage was calculated. The size of the pool and the FEC registers may, however, vary, depending on how the network is configured: the pool size of block elements for FEC and DTSN is the size of each element (48 bits) times the maximum number of blocks the network can be configured to handle. The FEC register size will depend on how large is the application layer payload.

$$RAMusage = PoolBlockQueueDTSN + PoolBlockQueueFEC + 2*FECRegister \quad (1)$$

To achieve and present a numeric result, we will consider that the maximum number of multimedia blocks is five and that the application payload size will not exceed one hundred bytes. The result of this balance will be the minimum RAM occupied by the implemented system for the simulation of section V.A using FEC.

Equation 2 presents the result of this RAM utilization balance. Although 260 bytes may seem insignificant compared to most nowadays applications, this value is not so low for wireless sensor networks software. It must be borne in mind that this is only the RAM usage for the developed differentiated reliability system, that of course, has to be added to application, basic transport, network, and link layer RAM usages before being installed into a wireless mote.

$$RAMusage = 5\times BlockQueue\ element + 5\times BlockQueue\ element + 2\times 100bytes = 5\times 48+5\times 48+2\times 100\times 8 = 2080\ bit = 260\ bytes \quad (2)$$

VI. Conclusion

A. Conclusions

DTSN's differentiated reliability service was implemented in order to increase system throughput and reduce network traffic, establishing for each multimedia block a minimum reliability grade.

The new DTSN service was simulated and the results obtained have shown that in most situations, the DTSN differentiated service significantly increases the throughput and decreases the overhead associated with non-critical traffic. Even in the rare situations where throughput decreases when lowering the reliability levels, additional retransmissions into the network are avoided, which is also an important goal from the points of view of energy-efficiency and interference. Moreover, placing unimportant network traffic in low reliability level multimedia blocks, allows higher levels of quality of service to be granted to critical blocks belonging to other multimedia streams crossing the network at the same time.

A packet recovery mechanism (FEC layer) proved to be working correctly, increasing the system throughput. The cross layer interaction between the FEC and transport layers allowed memory and processing time to be saved. Actually, this kind of interaction mechanisms are often used in wireless sensor networks programming. The layer concept disappears, being replaced by a functionality model where all the layers cooperate with each other to provide the necessary services to network traffic. This will leverage resources, increase network lifetime and provide high levels of traffic quality of service which is very important, especially for multimedia data transport.

B. Future work

Although the achieved results demonstrated the efficiency of the developed system, some new ideas arouse during the project implementation, which deserve to be presented here as basis for future work.

Intermediate nodes caching proved to be working efficiently, allowing most of the retransmissions to be local and not end-to-end. With the new DTSN differentiated reliability service, new caching methods may be developed in order to give higher reliability blocks higher caching priority. If most packets that belong to critical blocks are stored in intermediate nodes' cache, more retransmissions may be avoided, increasing the system transmission rate. Therefore, a cache management system must be implemented to avoid caching non-critical data in detriment of high reliability packets.

When several flows are active at the same time in the same WSN between different sender-receiver pairs, there is a probability that some nodes are crossed by more than one flow. In this case, there might be a tough competition for the scarce WSN node memory. In such scenarios, the caching probabilities should be dynamically set so that the service offered to the different flows is fair, while optimizing the overall quality of service (often contradictory goals). The development of distributed algorithms for the configuration of intermediate node caching probabilities is a challenging and yet largely unaddressed topic.

In order to allow the FEC layer to recover more than one packet per block, a more powerful erasure code must be implemented. That code must be able to retrieve more lost

block packets without increasing too much the block size. Additionally, the FEC code may also be used in intermediate nodes to enhance block reconstruction capabilities. Naturally, this has a disadvantage of increasing a lot the complexity of intermediate nodes that would have to do a lot more than caching packets.

In some WSN scenarios it may be advisable to use more than one route simultaneously for sake of load and energy balancing. In order to efficiently operate on top of a multi-path routing protocol, DTSN must keep some knowledge about the assignment of packets to routes, so that NACK packets may follow the reverse paths where there is the possibility of finding the lost packets in cache. This aspect of cross layer optimization of DTSN was left for further study.

REFERENCES

[1] Chonggang Wang, Kazem Sohraby, Bo Li, Yueming Hu, "A Survey of Transport Protocols for Wireless Sensor Networks". IEEE Network, Vol. 20, No. 3. (2006), pp. 34-40.

[2] Ian F. Akyildiz, Tommaso Melodia, Kaushik R. Chowdhury: "A survey on wireless multimedia sensor networks". Computer Networks, Vol. 51, No. 4. (March 2007), pp. 921-960.

[3] Bruno Marchi, António Grilo, and Mário Nunes, "DTSN – Distributed Transport for Sensor Networks", Proceedings of the IEEE Symposium on Computers and Communications (ISCC'07), Aveiro, Portugal, 2007.

[4] Joseph P. Macker, "Reliable Multicast Transport and Integrated Erasure-Based Forward Error Correction", MILCOM 97 Proceedings. November 5, 1997.

[5] TOSSIM, a TinyOS simulator . http://www.tinyos.net

[6] IEEE Std. 802.15.4, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", 2003

João Castro Almeida received his M.Sc. in Electrical and Computer Science Engineering from Instituto Superior Técnico, Technical University of Lisbon (IST/UTL), Portugal, in October 2008.

António Grilo received the Information Technology Engineering degree in 1996, the M. Sc. degree in Electrotechnical and Computer Engineering in 1998 and the PhD in Electrotechnical and Computer Engineering in 2004, all from the IST, Technical University of Lisboa, Portugal. He is Assistant Professor at IST, where he lectures subjects related with computer networks and telecommunications, in under-graduate and graduate courses. In 1995 he joined INESC, Lisboa, where he is currently a senior researcher. His current research areas are Wireless and Mobile Networks and Network Centric Military Communications. From 1996 until now he has been working in several European projects, namely ACTS projects ATHOC and AROMA and IST projects MOICANE, OLYMPIC, AIRNET and UbiSeq&Sens. He is currently working in IST project WSAN4CIP.

Paulo Rogério B. A. Pereira received his graduation, M.Sc. and Ph.D. degrees in Electrical and Computer Science Engineering from Instituto Superior Técnico, Technical University of Lisbon (IST/UTL), Portugal, in 1991, 1994 and 2003, respectively. He is an assistant professor of computer networks related subjects at IST/UTL, and a researcher at the Communications and Mobility Networks laboratory of INESC-ID Lisbon, since 1991. He has participated in the IST European projects EuroNGI, EuroFGI, EuroNF, UbiSec&Sens and WSAN4CIP. His research interests include IP quality of service, network management and wireless sensor networks.