Supporting Protocol-Independent Adaptive QoS in Wireless Sensor Networks

Evy Troubleyn, Eli De Poorter, Peter Ruckebusch, Ingrid Moerman, Piet Demeester
Ghent University - IBBT
Department of Information Technology (INTEC)
Gaston Crommenlaan 8, bus 201, 9050 Ghent, Belgium
Email: Evy.Troubleyn@intec.ugent.be

***Abstract*—Next-generation wireless sensor networks will be used for many diverse applications in time-varying network/environment conditions and on heterogeneous sensor nodes. Although Quality of Service (QoS) has been ignored for a long time in the research on wireless sensor networks, it becomes inevitably important when we want to deliver an adequate service with minimal efforts under challenging network conditions. Until now, there exist no general-purpose QoS architectures for wireless sensor networks and the main QoS efforts were done in terms of individual protocol optimizations. In this paper we present a novel layerless QoS architecture that supports protocol-independent QoS and that can adapt itself to time-varying application, network and node conditions. We have implemented this QoS architecture in TinyOS on TmoteSky sensor nodes and we have shown that the system is able to support protocol-independent QoS in a real life office environment.**

***Keywords*-Wireless Sensor Networks; Quality of Service; Architecture; Layerless Design**

I. Introduction

Next-generation heterogeneous wireless sensor networks are characterized by dynamic and time-varying (i) node capabilities and (ii) network conditions. For instance, sensor nodes can have heterogeneous resources in terms of energy, memory or computing capacity, and the nodes are often resource-constrained. Moreover, the wireless links can be very unpredictable due to fading, shadowing and the presence of mobile nodes.

Furthermore, these sensor networks may simultaneously support (iii) diverse applications, each of them having its own specific Quality of Service (QoS) requirements. For instance, the next-generation wireless sensor networks will not only be used for monitoring, tracking or building automation, but also for more challenging streaming applications such as voice and video. An example can be found in the large interdisciplinary IBBT-DEUS project [1] where academic partners, industry and non-profit organizations are collaborating on the development of a generic cost-efficient sensor network platform that can be easily deployed in diverse user scenarios. In one of the use cases, the sensor network will assist elderly persons, in particular persons with dementia, in and around residential care homes. As elderly people often stray, they are kept in an isolated wing of the care home today. The DEUS project aims to offer elderly person more freedom and mobility. To this end several services need to be deployed on top of the sensor network: (i) indoor (within the care home) & outdoor (garden and walking area around the home) positioning, (ii) emergency call realized through an alarm button on a sensor device carried by the elderly person, (iii) voice call between the elderly person and a nurse automatically established upon an emergency call or when the elderly person is located in an area where he/she is not allowed to go. These services have different requirements in terms of reliability, delay and bandwidth and the sensor network is expected to guarantee the different QoS requirements despite the dynamic nature of the wireless environments and the resource constraints (energy, bandwidth, memory) inherent to sensor networks.

This research paper will focus on QoS at an architectural level. Until now, most QoS efforts in wireless sensor networks are limited to the protocol level either through making network protocols QoS-aware or through cross-layer interactions. For instance, in [2], several QoS-aware MAC and routing protocols were proposed. However, they only focus on a few QoS parameters such as reliability [3] and delay [4], while other parameters such as jitter and bandwidth are ignored. Additionally, these protocols are tuned to a specific network environment or a specific application and also often ignore energy constraints. Since energy resources are scarce in wireless sensor networks, our architecture aims to deliver the right QoS guarantees while adapting itself to energy constraints. By treating QoS at an architectural level, QoS can be introduced irrespective of the applied network protocols and applications. This way, QoS support can be easily turned on/off and is transparent for protocol and application developers. Supporting protocol-independent adaptive QoS at an architectural level allows and simplifies time-varying global QoS optimization.

The remainder of this paper is organized as follows. In section 2, we give a short overview of the related work on QoS in networks. Afterwards, in section 3, we motivate the choice for a layerless architecture as starting point for our QoS architecture. Next, in section 4, we present our adaptive QoS architecture in general and discuss the main architectural building blocks. Section 5 will examine in more detail the fundamental protocol-independent QoS mechanisms of the architecture. In section 6, we will discuss the implementation of the QoS architecture on real sensor hardware in a real life environment. Some more advanced QoS mechanisms will be addressed in section 7 on future work. Finally, we will conclude this paper with a short summary of our work.

II. Related Work

QoS is in fact a very broad concept and is often defined as an objective measurement of the services delivered by the network expressed in terms of bandwidth, delay, reliability and jitter. However, Quality of Service (QoS) is different from Quality of Experience (QoE). Quality of Experience can be defined as the subjective measurement of the services delivered by the network as observed by the users. Sometimes, a good QoS can give a bad QoE and vice versa. An example of a QoE measurement is the mean opinion score (MOS) that expresses the quality of a voice call. Different codecs will give different mean opinion scores and each codec will be translated into different QoS parameters. Some codecs are for example more sensitive for packet errors and jitter. In the remainder of this paper, we will only focus on Quality of Service.

Existing QoS-oriented technologies for wired and wireless networks, such as ATM, MPLS, and IP, cannot directly be applied to wireless sensor networks. Although ATM [5] is very suited to support QoS, this connection-oriented technology is much too complex and not suited to use in fast-changing network environments as is often the case for wireless sensor networks.

IP is on the other hand connectionless but only offers best-effort traffic. However, the following QoS-oriented technologies are often used: IntServ [6] and DiffServ [7]. In IntServ, each flow can be treated individually which makes it very flexible. However, since every node has to maintain per flow state information, the drawback of IntServ is its scalability and the fact that it is too complex to use on sensor nodes with limited capabilities. DiffServ defines on the other hand some different service classes using the DSCP field. The advantage of this system is that the complex operations are moved to the edge routers, while the maintenance in the core routers remains simpler. The drawbacks are however that the quantitative information of each flow is lost after the aggregation in service classes and that in the case of wireless sensor networks even edge devices have limited capabilities.

MPLS [8] tries to combine the best of both the ATM and IP world but is still too complex to use on sensor nodes with limited resources.

Together with these QoS-oriented technologies, several QoS control and management solutions are available, going from simple over-provisioning to congestion avoidance, traffic shaping and resource reservation. Again, most of these solutions cannot directly be applied to sensor networks for reasons of network and node conditions and energy considerations. These challenging sensor network characteristics make it almost impossible to apply currently available end-to-end QoS solutions in wireless sensor networks.

III. Layerless Protocol Architecture

According to the OSI Reference Model [9], supporting QoS is one of the few network functionalities that is involved in all the system layers. Advantages of supporting QoS in such a layered structure are its standardized interface and its transparency. Many studies [10] argue that the layered protocol architecture, where each layer is designed and operated independently and which works very well for wired networks, seems to be very inefficient when deployed in wireless networks, which are much more dynamic and less predictable. Furthermore, this layered approach requires each layer to define its own QoS header information which gives much overhead on limited sensor nodes and the QoS capabilities of each layer are restricted by the QoS capabilities of the layer above and below.

Many cross-layer design ideas, going from the direct communication between layers up to complete layerless architectures, have already been presented [11], [12]. Direct communication between layers is the most straightforward way for cross-layer interaction and makes variables at one layer visible to the other (non-adjacent) layers at run-time. Another approach is based on a shared database, whereby the database can be regarded as a new layer, providing the service of storage/retrieval of information to all layers. The main challenges are to adapt the single layer protocols taking into account the information obtained from other layers and to design interfaces either between layers or between layers and a shared database. Yet another approach is joint optimization between adjacent layers. The disadvantage of such an approach is however its unclear structure and the inherent difficulty to deal with new layers in a plug and play manner.

In a layerless architecture, the protocols are organized in a modular way, enabling plenty of interactions between the different modules. While the latter approach offers the greatest flexibility, it is not anymore compatible with standard layered protocol stacks as is the case for the former cross-layer approaches. Layerless architectures may however be very effective for sensor networks, where devices have constrained memory, processing and battery capabilities. Recently E. De Poorter et al. have introduced a novel layerless protocol architecture, called information driven architecture (IDRA) [13]. The IDRA architecture is not based on packets, but on “information exchanges”. Protocols do not interact with packets, but only with the information they contain, meaning that protocols do not need to define complicated header structures and do not need to provide buffer spaces for storing the packets. Instead, packet creation and buffer provisioning are handled by the architecture. A conceptual representation of this architecture is shown in Fig. 1.

The main characteristics of IDRA are:

1) The system architecture is responsible for packet creation. *Protocols only have to hand over their information to the system*. This avoids exchanging redundant information, allows information aggregation and the protocols do not have to care about header creation.
2) In contrast to traditional systems, where each protocol has to store-and-forward packets, *IDRA has only one system-wide queue* where incoming packets are stored. The advantages are less processing overhead, less memory usage, simpler monitoring and management, and protocols can be kept simpler and smaller.
3) *Protocol logic and packet representation are decoupled by a packet facade*. Since the system is responsible for packet creation, it is very easy to use a different packet implementation (802.15.4, 6lowpan, own implementation, etc.) which allows compatibility with legacy systems. Protocols only have to ask the desired information (for instance the destination) which is independent of the used packet implementation.
4) *The system decides at run-time which protocols have to be used*. This pluggable protocol system makes it possible to dynamically change between different routing and MAC protocols.

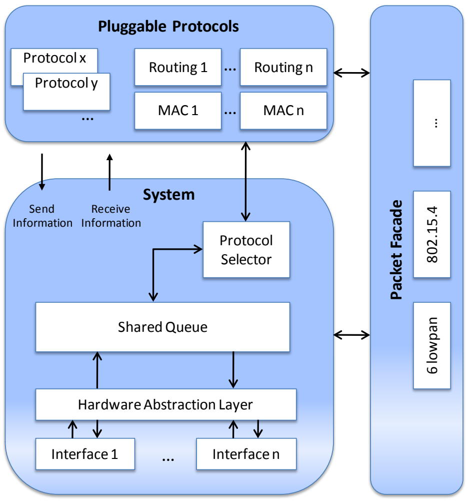

Figure 1. Information Driven Architecture: conceptual representation

IDRA is very suited to support QoS at an architectural level. The advantages are:

- *System-wide QoS*: since there is only one shared queue, information can be accessed, controlled and influenced at each network layer. This way, QoS decisions can be based on a global network view instead of a single layer protocol view.
- *Transparent QoS*: The packet facade and the information driven approach ensure that QoS information can be accessed in a transparent way. It makes it very easy to add protocol-independent QoS information such as a global priority level or protocol-independent QoS attributes such as the information reliability or the maximum allowed information delay.
- *Protocol-independent QoS*: Since there is no direct coupling between QoS and the network protocols, QoS can be simply enabled or disabled in the system depending on the application and user requirements. Basic QoS, such as packet priorities, can be enabled even if the protocols do not support any QoS features.
- *QoS-aware data-aggregation*: Since protocols hand over their information to the system, it is much easier to take QoS requirements into account for information-aggregation. The system has a global view on which information has to be routed to which destinations under which QoS conditions and can easily interact with the QoS parameters such as delay and reliability. In a layered architecture, QoS-based information-aggregation is almost impossible.
- *Heterogeneous QoS support*: The pluggable protocol system allows the QoS system to add new, more optimized protocols on per-need base. Based on the node's capabilities, network and application requirements, more or less QoS functionality can be plugged in or out the system.

This research paper explores the first three advantages. In future work, we will focus on QoS-aware data-aggregation and heterogeneous QoS support.

IV. Architecture

In this section, we describe how protocol-independent adaptive QoS can be provided at an architectural level in a layerless system approach. A conceptual representation is given in Fig. 2.

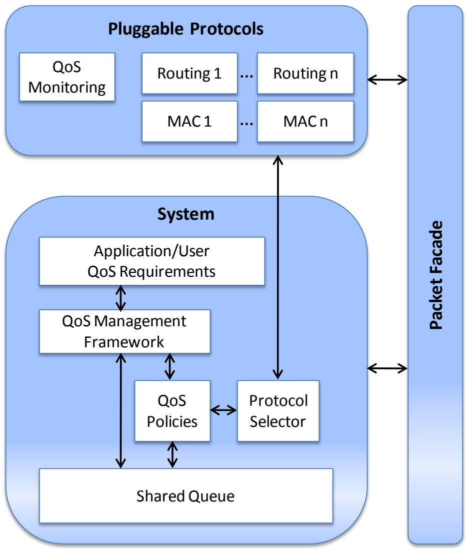

Figure 2. Supporting protocol-independent adaptive QoS in a layerless architecture: conceptual representation

The internal working of this protocol-independent QoS system is based on two packet-based QoS mechanisms:

- A mandatory packet *priority level*
- Optional additional *packet attributes*

Both mechanisms will be explained in more detail in section V. Until now, it is sufficient to know that all packet-based QoS interactions discussed in the following sections will be based on these two mechanisms. We will explain the QoS architecture based on a specific network scenario: an elderly monitoring scenario with two traffic flows: a reliable blood pressure monitoring application and an emergency voice call application.

A. Application/User QoS Requirements

Application requirements have to be translated into network requirements expressed in terms of end-to-end delay, reliability, bandwidth and possible jitter. For instance, the network's bandwidth will be the sum of the individual application bandwidths and the network's delay will be the most stringent application delay.

B. QoS Policies

The QoS Policies, shown in Fig. 2, define the internal rules for processing information through the system. As can be seen in Fig. 3, these rules are working on three levels:

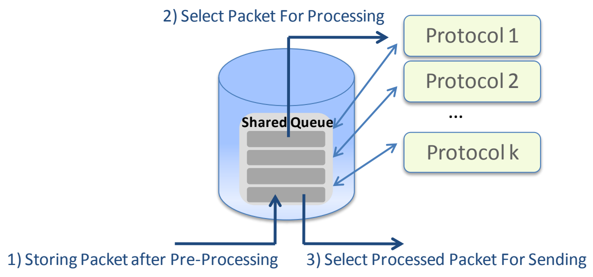
2) Select Packet For Processing
Protocol 1
Protocol 2
...
Protocol k
Shared Queue
1) Storing Packet after Pre-Processing
3) Select Processed Packet For Sending

Figure 3. Conceptual representation of the QoS Policies

1) When a remote packet arrives on a sensor node, packet-pre-processing can be executed. For instance, if the shared queue is almost full, a packet can be dropped in order to keep free spaces for new arriving packets. The QoS Policies will define which packet has to be dropped first: the reliable monitoring packet with the less stringent delay requirements or the voice packet with the lowest reliability level but with the most stringent delay requirements, or should it be a combination of both? The main advantage of the shared queue approach (from a QoS view) is that not only packets ready for pre-processing can be dropped, but even packets that are being currently processed or that already are fully processed and ready for sending.
2) When the system is ready to process a new packet for routing, we can select the packet to be processed in a QoS-aware way. Some packets, such as control and management messages always need to be processed first in order to keep the network alive. But for data packets, this packet selection process can be smartly controlled. At this stage, some minor modifications can be made on each packet. For instance, if our voice packet is processed, we can update the current delay that our packet already has undergone. Even the priority level can be changed at this stage. Suppose there are two voice calls with the same priority level. The maximum delay of the packets of the first voice call is almost reached, while the packets of the second voice call only have experienced a minor delay. In this case, the QoS Policies can decide to increase the priority level of the first voice call.
3) After a packet was processed, the control is again handed over to the shared queue where it stays until the MAC module is ready to send a new packet. Then, post-processing can select the most appropriate packet for sending. For instance, this decision can be based on the load and the sleep/awake duty cycle of the packet's next-hop. Again, the QoS Policies can influence these packet selection rules.

C. QoS Monitoring

QoS Monitoring can include three parts: Node Monitoring, Neighbor Monitoring and Network Monitoring.

The own *Node Monitoring* is responsible for monitoring the node's own load and energy, its own protocols and its radio capabilities.

Neighbor Monitoring on the other hand is responsible for monitoring one-hop neighbor information. Based on the node capabilities, more or less parameters can be monitored. Some examples are the load, the energy level and the duty cycle.

Network Monitoring is the part of the QoS Monitoring module that is responsible for monitoring (a part of) the general network load, the available network protocols on the other nodes and the QoS functionalities of the other network parts.

The monitoring information is used by the QoS Management Framework to intelligently change the rules from the QoS Policies.

D. QoS Management Framework

The QoS Management Framework (see Fig. 2) has several responsibilities.

Firstly, it is responsible for mapping the application QoS requirements to an initial packet QoS priority level. This can be statically done on design-time or more dynamical by using more complex learning techniques. Furthermore, it can add additional QoS attributes to the packets. For example, consider the scenario with the reliable monitoring application. In this case, a default priority level will be set together with additional information about the reliability.

Similarly, in the voice call scenario, a high priority level will be needed and extra packet-delay information can be added, for instance the maximum delay that such a packet is allowed to travel and the current delay already travelled until now. Initially, this current delay will be set to 0, but this value will be updated in the intermediate nodes, as explained in the QoS Policies section.

Secondly, the QoS Management Framework is responsible for controlling and managing the QoS Policies' rules. Based on some QoS Monitoring information and the currently available applications, it can be useful to change these rules. For instance, if our energy level becomes low, it could be better to only drop packets that are in the pre-processing phase instead of taking all the packets in the shared queue into account (and thus also the packets that are already fully processed).

V. Packet-based Adaptive QoS Mechanisms

When handling the different applications in wireless sensor networks, we also have to handle the different traffic/packet flows. Basically, we can make a distinction between a class-based and a flow-based approach.

In a flow-based approach, such as IntServ in IP networks, each traffic/packet flow is treated individually. This classification is very flexible, but has problems with its scalability because intermediate nodes have to maintain per flow state information.

In a class-based approach, such as DiffServ in IP networks, several service classes are defined. This approach is much simpler, but after aggregating the flows in classes, an individual traffic/packet flow can no longer be identified. Since sensor networks are used for many diverse applications deployed in dynamic environments on sensor nodes with limited resources, wireless sensor networks ideally need a combination of both:

- The simplicity and scalability of a class-based approach
- The flexibility of a flow-based approach

Therefore, our packet-based adaptive QoS mechanisms presented in this paper are based on a combination of both.

Since QoS in sensor networks has to be kept simple, a fixed amount of QoS classes is defined. These classes are called priority levels and determine the general behavior of a packet type. They are discussed in more detail in subsection V-A.

Because sensor networks can cover many diverse applications, it is very desirable to give the traffic flows a more individual character. Therefore, some extra attributes can be added to each flow. These attributes are used for fine-grained packet control, within the limits of the chosen priority level. The attributes are discussed in more detail in subsection V-B.

A. Novel QoS Priority Levels

Since sensor networks will support many diverse applications, each having their specific QoS requirements, it is not straightforward to fit each application in a predefined QoS class for processing and routing packets. For instance, a low priority packet can have a high reliability or, vice versa, a high priority packet can have a low reliability level. As a consequence, there is no one-to-one mapping available between a QoS class and a processing sequence.

We therefore propose a fixed amount of priority levels as QoS classes. To give each packet more flexibility, QoS attributes are added.

Table I. QoS Priority Levels

QoS Priority Level	**Description**
7	Reserved (MAC control information)
6	Reserved (Routing control information)
5	Reserved (Monitoring/Management information)
4	Real-Time traffic (critical mode)
3	Real-Time traffic (default mode)
2	Time sensitive traffic (critical mode)
1	Time sensitive traffic (default mode)
0	Best Effort traffic

As can be seen in Table I, eight priority levels are defined. The three highest priority levels are reserved for control, management and monitoring messages. These messages will always have the highest priority in order to prevent deadlock situations and thus keep the network alive.

Additionally, we see that both the time-sensitive traffic and the real-time traffic have two modes: a default mode and a critical mode. Initially, each data packet will have priority level 0, 1 or 3 based on the application QoS requirements. It is important to remark that these data characteristics are defined by the user/application, but that the translation into a priority level is made by the QoS Management Framework of the system.

The definition of two modes for the same traffic class can be justified by the following example. Suppose there are two simultaneous voice calls with initial priority level 3. The packets of the first voice call, while travelling through the sensor network, reach a delay close to the maximum delay. By allocation a higher priority, we give the packets of the first voice call a higher chance to still arrive on time. The priority of the second voice call, which has enough time left, remains unchanged.

B. QoS Packet Attributes

Since QoS priority levels are not sufficient to build a flexible QoS architecture, extra QoS attributes can be added to each packet (see Table II).

For instance, when considering the reliable monitoring application, a reliability-attribute can be added. Each network protocol supports this attribute to the best of its abilities. For example, the MAC module can choose to request acknowledgement messages, and the QoS Policies can decide not to drop a reliable packet.

Table II. QoS Packet Attributes

Attribute	**Description**	Required
Priority	Priority of network packets (see Table I)	Yes
Current_Delay	Travelled packet delay until now	No
Max_Delay	Max. allowed end-to-end packet delay	No
Reliability	Packet reliability indication	No

In addition, information about the current_delay and the maximum_delay can be added. This information can be used to drop packets if their deadline is already passed in order to not occupy the medium with unnecessary packets, or to allocate temporarily a higher priority level if the deadline is almost reached.

VI. Implementation and Evaluation

In the following, we will present a basic implementation of the QoS architecture. Currently, the Application/User QoS Requirements module is implemented as a database entity. Furthermore, the QoS Policies module and a basic version of the QoS Management Framework module are implemented. These modules allow a prioritized routing strategy with an intelligent packet dropping mechanism.

A. *W-ilab.t*

The implementation of the QoS architecture is done on TmoteSky sensor nodes using TinyOS 2.1.0 and nesC as a programming environment. Furthermore, we use the real life wireless testbed "W-ilab.t" [14].

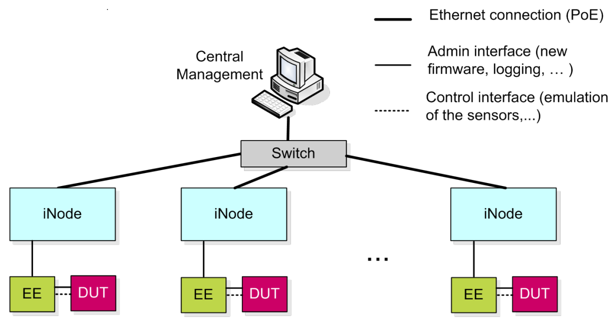
Ethernet connection (PoE)
Central Management
Admin interface (new firmware, logging, ...)
Control interface (emulation of the sensors,...)
Switch
iNode
iNode
iNode
...
EE
DUT
EE
DUT
EE
DUT

Figure 4. W-ilab.t Architecture

The sensor network part of this testbed contains 200 TmoteSky sensor nodes, spread on three floors in a 100m x 15m office building. Each of these sensor nodes is connected by an environment emulator (EE) and a small Alix computer (iNode), as can be seen in Fig. 4. The EE allows event emulation, e.g. sensor events or injection of audio. The iNodes are responsible for configuring the sensor node and are connected to a central management system [15].

B. *Experimental Setup*

In our experiment, half of the third floor of the W-iLab.t testbed is used. As a MAC protocol, a simple MAC protocol is used that checks periodically if it has packets to send. As a routing protocol, an own implementation of the Dynamic MANET On-Demand (DYMO) routing protocol is used.

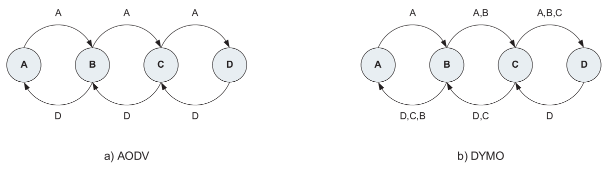

Figure 5. Comparison between AODV and DYMO

DYMO [16] is a reactive routing protocol similar to AODV based on route request (RREQ), route reply (RREP) and route error (RERR) messages. The main difference between DYMO and AODV is that DYMO can append additional intermediate node information (see Fig. 5), and that its somewhat simpler design makes it more appropriate for an implementation on resource-limited devices such as sensor nodes. Our modification to DYMO lays in the fact that we only forward the RREQ message if our receiver quality is higher than a certain threshold (1). That way, we prevent that the network becomes flooded by unnecessary RREQ messages.

$$Prob_{forwarding} = Prob_{receiving\_RSSI > Threshold} \qquad (1)$$

C. *Results*

In the following, the testbed results are shown in two scenarios. In the first scenario, two traffic flows with QoS support are considered: 1 high priority traffic flow and 1 low priority traffic flow. In the second scenario, two traffic flows without QoS support and thus with the same priority level are considered. For both scenarios, some throughput/drop results will be discussed.

1) Scenario 1: with QoS support: In this first scenario, two traffic flows with QoS support are considered. To illustrate this QoS support, two traffic flows with different priority levels are used. Both traffic streams will send a packet of 70 bytes payload every 150ms and each node checks every 100ms if it has a packet to send. At the beginning of the experiment, there is only 1 low priority traffic flow (flow 1) between sensor nodes 25 and 200. After a while, a high priority traffic flow (flow 2) is set up between node 24 and 200. As can be seen in Fig. 6 both flows meet each other at node 54. Since more packets arrive at node 54 than it can process, some packets will have to be dropped. The collected database results show that at the end of the experiment 2017 packets from the low priority traffic flow were dropped while 0 packets from the high priority traffic flow were dropped. These results are also shown in the left part of Fig. 8.

2) Scenario 2: without QoS support: In this second scenario, two traffic flows without QoS support are considered. In the QoS architecture, this scenario can be simulated by using two traffic flows with the same priority level. As in

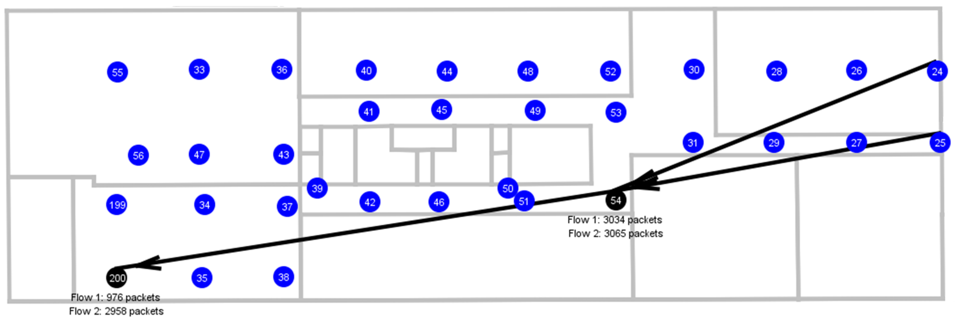

Figure 6. With QoS support: adding a high priority data flow

scenario 1, one traffic flow starts sending between node 25 and 200. After a while, the second traffic flow with the same priority level is set up between node 24 and node 200. This time, the collected database results show that 1098 packets from the first traffic flow were dropped while 1100 packets from the second traffic flow were dropped (Fig. 7). The right part of Fig. 8 shows these results.

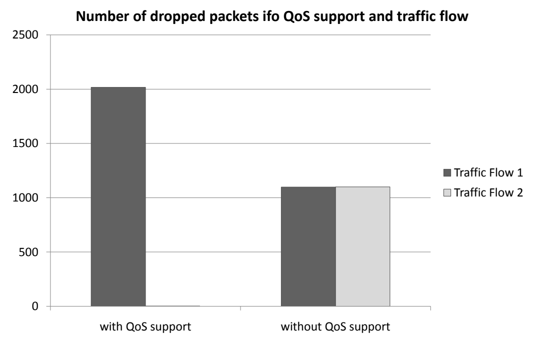

Figure 8. Number of dropped packets with and without QoS support

D. Memory Footprint

Table III. Memory Footprint

Module	ROM (bytes)	RAM (bytes)
IDRA system	20236	5344
Broadcast Routing	390	111
DYMO Routing	5008	312 (+18 per route)
Simple MAC protocol	844	24
Advanced MAC	7136	1264
Neighbor database	8536	2631
QoS Policies	1816	10
QoS Management Framework	3072	238
QoS Application Database	4068	668

Table III shows the memory footprint of the QoS modules compared to the memory footprint of the other IDRA system modules. One of the characteristics of IDRA is its low protocol memory cost, at the price of a somewhat bigger initial memory cost. When comparing the QoS system in IDRA using a simple MAC protocol and a broadcast routing protocol, the total QoS architecture takes about 29% of the total system's ROM memory and about 14% of the system's RAM memory. This is not negligible and it is the price we pay for a better overall QoS. However, if we compare our QoS architecture with an IDRA system with more advanced modules, for instance when taking into account DYMO routing in combination with a more complex MAC protocol with its own neighbor repository, the absolute QoS footprint remains unchanged, while the relative QoS footprint is decreased to 18% of the system's ROM memory and 9% of the system's RAM memory. We note that the implementation of the QoS architecture is protocol-independent.

VII. Future Work

Supporting protocol-independent adaptive QoS is only the first step towards a fully functional wireless sensor network QoS architecture.

In future work, we will investigate how QoS can transparently interact with network protocols such as MAC and routing protocols. Again, the layerless approach will be very useful. Imagine that the QoS Management Framework will be able to request the number of hops to a certain destination and can measure the end-to-end delay, it can use this information to calculate the per-hop delay towards that destination. Since the QoS Management Framework is aware of the maximum end-to-end-delay, more stringent parameter settings can be applied to the network protocols, e.g. temporary reduce the MAC sleep/awake duty cycle of the sensor nodes along the path to the destination.

The future QoS architecture will also have a smart mechanism for adding or replacing network protocols or adding other more advanced QoS modules. We could investigate which network protocol is best suited for a certain traffic type (reactive vs. proactive routing, contention based vs. slotted MAC), and we can change these protocols at run-time in order to gain a better overall QoS.

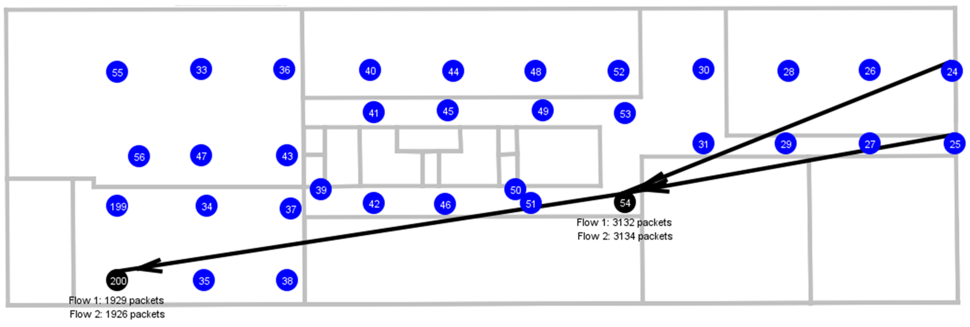

Figure 7. Without QoS support

VIII. Conclusion

In this paper, we have motivated why Quality of Service has to be supported in wireless sensor networks and we have presented how this could be done at an architecture level in a protocol-independent way.

The reason for taking QoS into account in wireless sensor networks was found in the inherent heterogeneous nature of wireless sensor networks. Since wireless sensor networks are subject to dynamic and time-varying node and network conditions and since they support many and diverse applications, adapting the network to the right QoS requirements becomes very important.

The presented architecture is able to support protocol-independent adaptive QoS in a layerless wireless sensor network architecture. The main system is based on a prioritized packet processing system that is controllable by a smart management system. In addition, the existence of several QoS attributes allows each flow to have a unique behavior, while keeping the implementation very simple.

We have implemented a basic version of this QoS architecture and we have evaluated the correct operation on a real life testbed with TmoteSky sensor nodes in an office environment. These initial results are very encouraging and, since QoS is an inherent part of the architecture, they are irrespective of the applied network protocols and applications.

Acknowledgment

The research of E. Troubleyn and E. De Poorter is funded by a Ph.D. grant of The Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). This research is also partially funded by the IBBT-DEUS project and by the FWO Flanders through projects 3G024310 and 3G029109.

References

[1] DEUS, "Deployment and easy use of wireless services," http://ilabt.ibbt.be/.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921–960, 2007.

[3] S. M.-E. Felemban, M.-C.-G. Lee, and M.-E. Ekici, "MM-SPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 738–754, 2006.

[4] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 39.

[5] D. E. McDysan and D. L. Spohn, *ATM: theory and application*. New York, NY, USA: McGraw-Hill, Inc., 1994.

[6] IntServ, "Integrated services," http://www.ietf.org/rfc/rfc1633.txt.

[7] DiffServ, "Differentiated services," http://tools.ietf.org/html/rfc2475.txt/.

[8] MPLS, "Multi protocol label switching," http://tools.ietf.org/html/rfc3031.txt/.

[9] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425 – 432, April 1980.

[10] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *Communications Magazine, IEEE*, vol. 43, pp. 112–119, 2005.

[11] S. Kota, E. Hossain, R. Fantacci, and A. Karmouch, "Cross-layer protocol engineering for wireless mobile networks: Part 1," *IEEE Communications Magazine*, vol. 34, no. 12, pp. 110–111, 2005.

[12] S. Kota, E. Hossain, R. Fantacci, and A. Karmouch, "Cross-layer protocol engineering for wireless mobile networks: Part 2," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 85–136, 2006.

[13] E. DePoorter, I. Moerman, and P. Demeester, "An information driven sensornet architecture," in *Proceedings of the 3nd International Conference on Senor Technologies and Applications*, Athens, Greece, June 2009.

[14] iLab.t Wireless Lab, "W-ilab.t," http://www.ibbt.be/en/project/deus.

[15] L. Tytgat, B. Jooris, P. D. Mil, B. Latre, I. Moerman, and P. Demeester, "Demo abstract: Wilab, a real-life wireless sensor testbed with environment emulation," in *European conference on Wireless Sensor Networks, EWSN adjunct poster proceedings (EWSN), Cork, Ireland*, February 2009.

[16] DYMO, "Dynamic manet on-demand routing protocol," http://tools.ietf.org/html/draft-ietf-manet-dymo-17/.