Session 3 – Wireless Networks

Chair: Lucia Lo Bello

Rapporteur: Christos Koulamas

Session Summary: Wireless Networks

Rapporteur: Christos Koulamas
Industrial Systems Institute
Patras, Greece
koulamas@isi.gr

1. Introduction

The focus of the third session of the 5th international workshop on real-time networks was on the specifics of wireless networks.

Three papers were presented, covering a) an approach to dynamically calibrate the data transfer in wireless sensor networks between pushing and pulling [1], b) a usage of a prioritized MAC protocol to efficiently compute simple or complex aggregated quantities [2], and c) a proposal of practical service differentiation mechanisms in order to improve the performance of the slotted CSMA/CA operation of 802.15.4 for time-critical events [3].

Chairing this session was Lucia Lo Bello from the University of Catania in Italy.

2. Adaptive Leases in Wireless Sensor Networks

The work described in the first presentation was on the selection between a push or pull mechanism for the data transfer from a source to a sink. Each mechanism has its advantages and drawbacks for each side of the data transfer. With the objective to maximize the lifetime of the system, the authors propose a dynamic scheme, called “adaptive leases”. According to this, it is possible to negotiate the preferred mechanism dynamically at run-time, depending on the current state of the source and sink in order to save power at the critical side.

The authors define specific lease strategies for each possible major objective and they encapsulate these in a special network layer. The definition of a layer hides the internals of the “adaptive leases” scheme from the applications, which, in turn, access the data transfer functionality through a simple API without the need to know whether the actual transfer will be completed in a push or a pull way. In the simulations performed, for an 802.15.4 based network and for a specific example, the results show an increase in the system lifetime by a factor of 2.

The discussion triggered by the questions after the presentation, was mainly on the relationship of the 802.15.4 internals with the proposed scheme, especially, on how this scheme may be combined with the power management mechanisms of the 802.15.4 standard and compared with the results of the usage of a PAN coordinator. The presenter clarified that their proposal fits better on a peer-to-peer network topology (i.e. ad-hoc). After this, there was a proposal for a study of the presented mechanisms over other MAC protocols which may be more suitable in event driven architectures. Furthermore, the issue of scaling was discussed where it was made clear that the simulation studies were not focused on large scale wireless sensor networks but on the validation of the energy savings due to the “adaptive leases” scheme in smaller device communities.

3. Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities

The second presentation dealt with the problem of computing aggregated quantities over a number of multiple values coming from multiple individual sensor readings in a wireless sensor network.

The authors propose an innovative algorithm for the computation of simple or complex aggregated quantities, by utilizing the special characteristics of a prioritized MAC protocol for the wireless medium access. Key aspects of such a protocol are a) the availability of a very large range of priority levels, b) the guarantee that it is collision-free, if priorities are unique, and c) the assumption that it is a dominance protocol, operating in a way similar to the CAN bus, but in a wireless medium.

The proposed algorithm makes feasible to compute simple aggregated quantities, as the minimum and maximum of a set of proposed values, with a time complexity which does not depend on the number of nodes, and which only increases very slowly as the possible range of the value increases. Moreover, the computation of more complex quantities, as the median of the set, can be also achieved with an independent to the number of nodes time complexity, but trading off

accuracy, since the solution is based on an estimation of the actual number of nodes in the network.

The related discussions evolved in two main axes. The first one was on what happens in the case of a network with a small number of nodes, considering the computation of a complex quantity, like the median, which is actually an estimation. The remark was that the estimation error increases when the number of nodes decreases, since the estimation is based on the intuition that if there is a large number of nodes which propose a randomly generated number, then the minimum random number will be very small.

The second axis of the discussion was triggered by practical issues related to the realization of the proposed prioritized MAC, in order for the proposed solution to bring a real advantage. The facts are that with the currently available technology, the prolonged duration of the transmission of the priority field may not always lead to better absolute time figures for the distributed computation, compared to the usage of a standard MAC and the usage of a typical algorithm which uses '*m*' messages for a network with '*m*' nodes and for typical values of '*m*'.

4. Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks

The third and last paper presented in the Wireless Networks session contributes towards the provision of service differentiation mechanisms to the 802.15.4 protocol, in order to improve its performance for time-critical events.

The authors suggest the proposed differentiation mechanisms to be based on the provision of different sets of values for selected parameters of the medium access algorithm for each differentiated service class. They propose a two-priority scheme, with the low priority assigned to normal data frames and the high priority to time-critical information exchange, such as GTS allocation requests, alarms, PAN management commands etc. The realization of the two priority classes is based on different values for the contention window (CW), the minimum backoff exponent (macMinBE) and the maximum backoff exponent (aMaxBE) parameters of the 802.15.4 MAC operation.

The results extracted from the simulation of four different scenarios, in both a FIFO and a priority based queuing show the efficient differentiation on the selected performance metrics for the two service classes, in all scenarios referring to a fully connected network; an achievement which comes with no significant changes but only minor add-ons to the current version of the 802.15.4 standard.

During the discussions on the presented results, clarifications were given on the relationship and usage of the changes in the sensing and receiving sensitivity as a way to produce a partially connected network and study the effects of the hidden node problem. A case where the service differentiation is actually degraded but a case also were the selection of priority queuing combined with large values for the BE range parameters leads to a more energy efficient operation. Moreover, similarities in the approach have been noticed to exist in other research efforts, dealing with QoS provision to the WLAN standards, prior to 802.11e, which could be valuable sources for a qualitative cross evaluation of the results or for leading to more ideas applicable also to the 802.15.4.

5. Concluding Remarks

The quality of all presentations fed a number of equivalently interesting discussions on various energy and real-time performance issues in wireless networks and their applications. It is noticeable though that all three papers focused on specific aspects of wireless sensor networks and their related protocols, even if the session title did not put such a constraint; an indication of the increasing importance of the area and of the growing attention it receives from the research community.

References

[1] Robert van Herk, Willem Fontijn, "Adaptive Leases in Wireless Sensor Networks". *Proceedings of the 5th International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

[2] Bjorn Andersson, Nuno Pereira and Eduardo Tovar, "Using a Prioritized MAC Protocols to Efficiently Compute Aggregated Quantities". *Proceedings of the 5th International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

[3] Anis Koubaa, Mario Alvez, Bilel Nefzi and Ye-Qiong Song, "Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for the Time-Critical Events in Wireless Sensor Networks". *Proceedings of the 5th International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

Adaptive Leases in Wireless Sensor Networks

Robert van Herk
Philips Research Eindhoven
robert.van.herk@philips.com

Willem Fontijn
Philips Research Eindhoven
willem.fontijn@philips.com

Abstract

A basic question in network communication is whether to push or pull data. Sometimes it can not be decided beforehand which mechanism to use. This article discusses an approach that makes it possible to dynamically calibrate between pushing and pulling in wireless sensor networks. This approach is called 'adaptive leases'. It allows for the sources and sinks of sensor data to decide for each communicated value at run-time whether pulling or pushing will be used. Doing so, nodes can dynamically optimize for power, memory, and bandwidth usage.

1 Introduction

We can divide the mechanisms to send data over a network into two categories: push and pull.

In a pushing network, a data *source* sends updates to its *sinks*. Usually the sources keep an administration on which sinks are interested in what data. Another solution is to broadcast all changes over the network, although in multihop networks the corresponding network load may be unacceptable.

In a pulling network, the burden of getting fresh data is put upon the sinks. The sink usually polls the source at a certain interval so that it is notified of changes when it is ready to receive them.

Table 1 describes some typical features. Beneficial features are marked with a plus sign, and disadvantages with a minus sign.

This article describes a solution in which, for each communicated value, the sources and sinks can negotiate and calibrate between pushing and pulling dynamically. We call this approach *adaptive leases*. The approach is most applicable in truly wireless sensor networks, i.e. networks with only battery operated nodes. One may think of e.g. wearable and in-body applications, for example in the medical domain.

Push
+ Source does not need to listen for incoming requests
- Sink must listen continuously to handle incoming updates
- Source must remember which sinks are interested in its data
+ Sink can be stateless
+ Updates are sent to the sinks immediately
- For every event there will be a message even if the sink is not interested at that moment
Pull
- Source must listen continuously to accept incoming pull requests
+ Sink does not need to listen when it is not interested in updates
+ Source can be stateless
- Sink must remember where to get which data
- Updates are sent to a sink only after it polls for fresh data
+ There will be no message when the sink is not interested at that moment.

Table 1. Push vs. pull

2 Related work

Similar approaches are already used in TCP/IP networks. For World Wide Web caches, Duvvuri et al. have described a system that uses adaptive leases [3]. A similar system is used by Microsoft to synchronize data between mobile devices and a computer [7]. It has been shown that smart mechanisms that combine pulling with pushing behavior in some general applications outperforms pure push [4].

There are algorithms designed to work with so-called *standing queries*. For example, in the APTEEN algorithm [6] the sink sends in its request to the source under what conditions new data is desired. The source then sends an update only if this condition is met. Adaptive leases can be used well together with this approach, as will be shown in Section 7.

Also, TinyDB has an standing query mechanism [5].

This is somewhat similar to our approach, where the lease time is the time window. However, in TinyDB the source has no mechanism to negotiate whether to use a lease or not. It can only refuse to execute the query, terminate it prematurely or comply. With adaptive leases the sink can indicate why it wants or does not want a lease, and will be informed about the actual window the source will honor. The application can then anticipate this. Another important difference is that in adaptive leases, a separate adaptive leases layer hides the leases mechanism from the application programmer (see Section 6.2).

3 Adaptive Leases in WWW caches

The system Duvvuri et al. describe [3] works as follows: When a WWW cache decides to store a copy of a web page, it sends a special message to the web server requesting a *lease* on that web page.

A lease can be thought of as a contract where the source (in this case a web server) has the obligation of pushing fresh data to the sink (in this case a web cache) for the duration of that contract. This duration is called the lease period. Once the contract will terminate, the sink must send a lease renewal request to the source. In the system of Duvvuri et al., the source decides on the lease periods. It may also decide that it does not want to engage in a requested lease at all, in which case the lease is denied.

This mechanism allows the web server to dynamically calibrate between pushing and pulling. For example, a web server with lots of outstanding leases may decide to use short lease times in new leases. In [3], this is called "Lease Duration Based on the State Space Overhead". In other cases, the web server may decide to grant a lease with a long lease period, so that the amount of control messages regarding the leases is minimized. This is called "Lease Duration Based on Control Message Overhead".

The shorter the lease times, the more the network will work as a pull system. When a sink gets responses with lease durations of zero, that value is effectively pulled by the sink, yielding a control message for each request. When the lease time is infinite, the value is effectively pushed by the source. But intermediate values are of course also possible; as such adaptive leases allow each sink and source to calibrate for each value they share between pushing and pulling.

4 Examples

As mentioned, we consider a topology in which resource constrained, battery operated sensor nodes are interconnected. In this topology, no mains-powered devices exist that have no power considerations and can afford to stay on constantly to relay messages.

4.1 Saving power at sink

We consider a patient temperature monitoring system. It consists of a thermometer and an actuator that shows the patient's temperature. Both nodes are carried with the patient and hence are battery operated.

In this example, the calibration between pushing and pulling is used to trade accuracy of the data at the sink against power usage at the sink. In order to show the current temperature, pushing is best used: the thermometer pushes a fresh value to the actuator immediately when the temperature changes. In order to achieve this, the thermometer grants the actuator a lease for the data. In a real-time system, this implies that the actuator must stay awake to deal with incoming updates immediately.

However, when the actuator's battery is almost depleted, a pulling system will be more appropriate, so that the device can sleep between pulls in order to save power. The actuator can then terminate the lease and pull a fresh value when the user indicates he wishes to see a fresh value, by pressing a button. The remaining time, the actuator can sleep to save power. Again, since we are dealing with a real-time system, the thermometer has to stay awake in order to deal with incoming requests immediately.

4.2 Saving power at sources

The opposite approach, i.e. granting leases, can be used to save power at the source.

Consider an application similar as the one explained above. The actuator may indicate it wishes to pull, for example because its battery is almost depleted. However, the source may have more stringent reasons *not* to allow pulling and grant a lease anyway, as its battery condition is more severe. In such a case, adaptive leases help in stretching the life-time of the source's battery. This is especially convenient in applications in which it is easier to replace or recharge the battery in the actuator (sink) than in the source.

This may be the case if the source is placed inside a patient. In such case, we want to delay the depletion of the source's battery. You may think of a slow release medicine system with an implanted sensor to measure sugar levels in blood and an externally accessible pump that can release insulin as required. The patient can replace the pump's battery by himself, but needs help from an expert to replace the sensor's battery. If the pump's battery is almost empty, which may happen while the patient is outside, it will start pulling data from the sensor and indicate to the patient that he needs to replace the battery. However, if the source's battery is almost empty too, it will always grant a lease, as replacing the sources battery requires an expert.

Another example is an application with sensors woven into clothing for monitoring a person during sports. It has a

carried read-out device that is easily recharged and multiple sensors that are more difficult to recharge. During outdoor sporting activities, the system would normally push to save power at the sources, and only switch to pull if the battery in the read-out device is getting empty to ensure the applications runs for the entire sporting activity. If the batteries of the sensors are, however, also almost depleted, they will grant a lease anyway, as replacing them is more difficult.

5 Lease strategies

If we use adaptive leases in wireless sensor networks, we can dynamically calibrate between the typical attributes pull and push systems have that are given in table 1. For wireless sensor nodes, this is especially important since wireless nodes typically have limited resources. The ability to dynamically trade memory footprint, required processing power and bandwidth usage between the sources and the sinks is beneficial. The opportunity to let battery operated nodes sleep if the batteries are running low can extend their up-time.

The sources and sinks can use the following strategies to calibrate:

- For reducing the number of unnecessary updates: sinks that receive a lot of update information but only are interested in a new value seldomly, e.g. only when the user explicitly requests it, can terminate their leases and start pulling. For reducing state overhead at a source the same approach can be used: sources that want to save memory by diminishing their administration on active leases can terminate leases and let the sinks pull. On the other hand, if we want to reduce the number of messages per update: sinks that constantly need fresh data can request leases.

- To save power at sink: if a sink wants to sleep, it should terminate all leases. Doing so, it will know not to receive updates unless it pulls for them, so it can go to sleep, only to wake up once it is interested in new data again. For the corresponding sources, it is compulsory to stay awake. Note that this strategy is used in the example given in Section 4.1.

- To save power at source: if sources want to sleep, they need to grant all leases. Once all leases are granted and if the source knows no other sinks remain that are interested but did not yet subscribe, it can safely go to sleep and only wake up to send updates. It is compulsory for the sinks of such a source to stay awake. This strategy is used in the example given in Section 4.2.

6 Our approach

Our approach is to define a network layer that negotiates, establishes and calibrates the adaptive leases, and then to give that layer such an interface that this functionality is hidden from the application programmer.

6.1 Adaptive leases layer

The new layer will do the following: when a sink is interested in a value and does not have a lease yet, it sends a message to the source[1]. The message contains:

- What value the sink is interested in.

- A value indicating whether the sink would prefer a lease, e.g.:

  - NoLease: just send the value once
  - Lease: send new values from now on. Optionally, we can include a desired lease termination condition determining for how long, or for how many updates, the lease should preferably last.

- A value containing the argumentation for indicated preference. This value is used for the negation. For example, if NoLease is indicated, the argumentation, sorted from strongest to weakest, may be an indication that:

  - ”Sink is almost out of battery power so cannot afford to stay awake”
  - “Sink is only interested at this moment”

  If Lease is indicated, the argumentation may be an indication that:

  - “Sink must have the value real-time”
  - “Sink would prefer to have the value real-time”
  - “Sink expects to need fresh values often”

When the source receive such a message, it will consider the preference and argumentation of the sink and it’s own state to decide whether to engage in a lease or not. Every message regarding an update of a value the source sends to a sink, will contain:

- The value

- Lease status, e.g.:

[1] Some discovery mechanism needs to be in place to find the address of the source; broadcasting or some central look-up system could be used. However, this is not in the scope of this article.

- NoLease: sink will have to send a new pull request if it is interested in the value again. This reply is used if the sink indicated in the request that it wants 'NoLease' and the source did not have stronger arguments to do engage in a lease. Also, this reply is used if the sink did indicate a preference for a Lease, but the source has stronger arguments not to engage. Finally, this value will be sent after the lease has expired.
- Lease: sink should not send new pull requests for this data. Instead, the source will push new values. Optionally, we can include the lease termination condition defining for how long, or for how many updates, the source expects the lease to last. This reply is used if the sink indicated in the request that it wants a 'Lease' and the source did not have stronger arguments not to engage. This reply is also used if the sink did indicate a preference for NoLease, but the source has stronger arguments to do engage in a lease.

6.2 Making leases transparent

To hide the above protocol from the application programmer, the adaptive leases layer needs to be totally transparent. To achieve this, we define an interface to operate the adaptive leases layer from the application layer using the following primitives:

- `subscribe(valueID)`
- `unsubscribe(valueID)`
- `publish(valueID, value)`: used by the sources.

Using this abstraction, the application programmer can subscribe the sinks to the values of interest in a natural way. He does not notice whether a lease is used or not: the layer should be implemented such that when the strategy prevents the sink from requesting a lease (e.g. because it is currently low on power), or when the sources refuses the lease, it automatically reverts to repeated pulling.

The desired strategies for requesting and granting leases (see Section 5) are implemented in the adaptive leases layer. One possibility is to implement the layer once, and make it configurable, so that some nodes can use the strategies to save power, whereas others can save on memory, and so on. Another possibility is to make multiple implementations, one for each strategy, and install one with a suitable strategy on each node.

7 Simulation

We have implemented the adaptive leases protocol in Java to show its feasibility. The implementation simulates SAND (Small Autonomous Networked Devices) nodes of which the application layer uses the above interface to subscribe and unsubscribe to data, and of which the adaptive leases layer uses the protocol described in this article. SAND nodes are very small wireless sensor nodes. They contain a Philips CoolFlux DSP, and a ChipCon CC2420 radio, which uses IEEE 802.15.4 [1] as communication protocol. One can attach multiple sensors and actuators to this core. We assume that each SAND node has a battery that contains about 10000 Joules when fully charged.

All cases mentioned in Section 4 are suitable for simulation, but for simplicity, we chose to simulate the following: node v is a wireless thermometer and node w an actuator that shows the temperature as measured by v. Both v and w are SAND nodes. Nodes v and w can send messages to each other.

The 802.15.4 MAC allows to use network coordinators that are essentially powered nodes that relay messages. As mentioned above, adaptive leases are most suitable for a fully wireless network, meaning that we cannot depend on other nodes to relay messages. In 802.15.4 this kind of communication is called *ad hoc communication*. In this setup, once a wireless node sends out data to another node, the other node must be listening in order to receive it. Otherwise the message will be lost.

Here we will demonstrate calculations that show the energy savings adaptive leases can accomplish in wireless sensor networks. We will compare the results of pure-pushing, pure-pulling and adaptive leases.

We assume the network to be more or less dedicated for our application. Therefore, we do not take collisions into account for traffic between v and w. If we would take collisions into account, the energy needed per message sent or received will increase [2], but since in this example most energy is spent on listening this will not significantly change the result.

7.1 Power consumption

7.1.1 Processor

In the SAND node, the CoolFlux DSP uses about 2 mW when turned on. When it is sleeping for a predetermined amount of time, it uses about 0.01 mW, running a low frequent oscillator to count down the waiting time.

Most of the time we will neglect the power used by the processor. When the device is sleeping however, meaning the radio is off and the processor is using 0.01 mW, we will take these costs into account. This is because the durations of the sleeping sessions are considerable.

7.1.2 Radio

When the CC2420 radio is in transceive mode, it consumes 30 mW. In receive mode, it consumes 35 mW. When it is idle, but its oscillator is running, it consumes about 0.712 mW [2]. However, because of inefficiencies in the current implementation of the SAND node[2], these numbers raise to 56 mW, 65 mW, and 1 mW respectively.

From the 802.15.4 standard [1], we calculate that sending a message with a payload of 10 bytes takes 0.7 ms. It takes 1 ms to start up the radio. Together, this means sending a message costs $0.7 \times 0.056 + 1 \times 0.001 + 1.7 \times 0.002$, hence, about 0.04 mJ. The terms are for sending, starting up the radio, and the processor, respectively. Note that the costs of the processor and of starting the radio are indeed negligible.

To pull for a new sample, we need to send a message requesting the data, which will take about 0.7 ms. Then, we need to wait at least one round-trip time before the answer arrives, and finally receive the answer. If the round-trip time is 50 ms, this means pulling a new sample costs $0.04+(50+0.7) * 0.065$, hence 3 mJ.

7.1.3 Thermometer

For measuring the temperature, v uses an analog sensor that measures a value that is transformed to a binary number. We set it to consume 100 mW and to take 10 ms to take a sample. These components will only be powered once taking a sample, which will happen one time per minute. This means that, on average, this component consumes 0.015 mW.

We use APTEEN [6] to send only necessary updates: we calculate how much the new sample differs from the previous value sent to w and only send an update if the temperature has changed more than 1 Kelvin. We choose that this is in average the case once every three samples, so once every three minutes. This means that when new values are pushed, on average one sample per three minutes is sent, which costs $\frac{0.04}{180} = 0.0002$ mW on average.

7.1.4 Actuator

For the actuator w, we assume that an array of LEDs is used to indicate the current temperature. The user can press a button, after which one LED will light up for ten seconds, indicating the current temperature. We assume the power of the LED is 5 mW. We assume that on average the user requests one temperature indication per hour. This means that, on average, this component consumes 0.013 mW.

In a pulling scenario, the device pulls for a new sample every minute, which hence costs $\frac{3}{60} = 0.05$ mW on average.

[2]This is due to a linear voltage regulator converting from 3.3 Volts coming from the battery to 1.8 Volts used in the CC2420

7.2 Pure-push

In pure-push, v sleeps all the time, except when taking or sending a new sample. However, w is constantly awake and ready to receive.[3]

This means that in this scenario v consumes $0.01 + 0.0015+0.0002 = 0.0117$ mW and w consumes $0.013+65$ mW.

This means that in this setup, v can run about 27 years, on a single battery. However, w can only run about 42 hours, on a single battery. This means that the total up-time of the system is 42 hours; after this time the user has to replace a battery.

7.3 Pure-pull

In pure-pull, v is constantly awake and ready to receive incoming pull requests, but w sleeps all the time, except when requesting a new sample.

This means that in this scenario v consumes $0.015 + 65$ mW and w consumes $0.01 + 0.013 + 0.013$, which is about 0.036 mW.

In this setup, v can only run about 42 hours on a single battery, whereas w can run for 9 years on a single battery. Again, the total up-time of the system is 42 hours; after this time the user has to replace a battery.

7.4 Adaptive Leases

Intuitively, adaptive leases will help because it can make the nodes switch between pulling and pushing, which is useful for synchronizing the pace at which the batteries of both v and w deplete.

7.4.1 Lease negotiation

We assume the effort of replacing the the battery in the thermometer is equal to the effort of replacing the actuator's battery. Therefore, we would like both batteries to be depleted at the same time.

We chose the following negotiation conditions:

1. If the battery of v has less energy than the battery of w, v always gives w a lease
2. Otherwise, no lease is granted and pulling must be used

In the messages from w to v, w includes the amount of energy left as argument to request a lease or not. When

[3]Of course, in a pushing scenario, we cannot let w sleep and only wake up once the button for a temperature indication is pressed by the user, because v will at that moment most likely be sleeping and not send a fresh value immediately.

v receives the request, it checks whether it can agree with w's proposal according to above negotiation conditions and decides whether to grant a lease or not.

7.4.2 Overhead of the protocol

We set the lease duration to 30 minutes, after which the lease is terminated and negotiation will take place again. Once the lease has terminated, v sends a message to w telling it that the lease has terminated and will have to wait at least one round-trip time for a lease renewal request. This costs v about 3 mJ per 30 minutes, which is 0.0016 mW. Also, w will have to send a lease renewal request once the lease has terminated, which costs 0.04 mJ per 30 minutes, which is negligible.

7.4.3 Results

It is easy to show that using adaptive leases leads to much longer up-time of the system. Both batteries will now deplete at the same pace. Running the adaptive leases protocol costs v an extra overhead of 0.0016 mW, which means its total energy consumption is now $0.0117+0.0016 = 0.0133$ mW when leases are granted.

To calculate up-time, we must to solve the following set of equations:

$$\begin{aligned} T_{push} + T_{pull} &= T_{total} \\ \frac{0.0133}{1000} \times T_{push} + \frac{65}{1000} \times T_{pull} &= 10000 \\ \frac{65}{1000} \times T_{push} + \frac{0.036}{1000} \times T_{pull} &= 10000 \end{aligned}$$

Hence, T_{push} and T_{pull} are both about 153000 seconds, which is about 42 hours. The total up-time is hence 84 hours.

This means that the up-time using adaptive leases is 84 hours, after which the user needs to replace both batteries. Effectively, adaptive leases doubled the up-time of the system.

8 Conclusion

Using adaptive leases is a promising mechanism for certain wireless sensor networks. It allows for nodes in the network to dynamically calibrate memory, bandwidth, processor and power usage. Contrary to standing queries, in adaptive leases the sink and the source negotiate on whether to engage in a lease, and possibly on the lease duration. Furthermore, the adaptive leases mechanism can be made totally transparent to the application programmer.

Adaptive leases seem to be most applicable in purely wireless sensor networks, such as wearable or in-body networks. Our simulation has shown that for some of these applications, adaptive leases can double the up-time of the system.

References

[1] IEEE Standard for Information technology – Telecommunication and information exchange between systems – Local and metropolitan area networks – Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Computer Society, Oct. 2003.

[2] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene. Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 196–201, Washington, DC, USA, 2005. IEEE Computer Society.

[3] V. Duvvuri, P. Shenoy, and R. Tewari. Adaptive Lease: A Strong Consistency Mechanism for the World Wide Web. Technical Report UM-CS-1999-041, 1999.

[4] M. S.-F. A. D. Leung. Pull vs. Push: A Quantitative Comparison for Data Broadcast. *"GLOBECOM -NEW YORK-"*, VOL 3:1464–1468, 2004.

[5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[6] A. Manjeshwar and D. P. Agrawal. APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 48, Washington, DC, USA, 2002. IEEE Computer Society.

[7] A. Yakhnin. Implementing Always-Up-To-Date Functionality in the .NET Compact Framework by Using Web Services. *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/autd_functionality_netcf_webservices.asp*.

Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities

Björn Andersson, Nuno Pereira and Eduardo Tovar
IPP Hurray Research Group
Polytechnic Institute of Porto, Portugal
{bandersson,npereira,emt}@dei.isep.ipp.pt

Abstract

Consider a distributed computer system such that every computer node can perform a wireless broadcast and when it does so, all other nodes receive this message. The computer nodes take sensor readings but individual sensor readings are not very important. It is important however to compute the aggregated quantities of these sensor readings. We show that a prioritized medium access control (MAC) protocol for wireless broadcast can compute simple aggregated quantities in a single transaction, and more complex quantities with many (but still a small number of) transactions. This leads to significant improvements in the time-complexity and as a consequence also similar reduction in energy "consumption".

1. Introduction

It has been recently discussed [1] that sensor networks often take many sensor readings of the same type (for example, temperature readings), and instead of knowing each individual reading it is important to know aggregated quantities of these sensor readings. For example, each computer node senses the temperature at the node and we want to know the maximum temperature among all nodes at a particular moment.

This can be solved with a naïve algorithm; every node broadcasts its sensor reading and hence all nodes know all sensor readings and then they can compute the aggregated quantity. This has the drawback that in a network with m nodes, it is required that m broadcasts are made. Considering that sensor networks are designed for large scale (for example thousands or millions of nodes), the naïve approach can be inefficient with respect to energy and cause a large delay.

In this paper we show that a prioritized MAC protocol for wireless broadcast can significantly improve the time-complexity for computing certain aggregated quantities. In particular we show that the minimum value can be computed with a time complexity that does not depend on the number of nodes. Also the time complexity increases very slowly as the possible range of the value increases. The same technique can be used to compute the maximum value. We also show how to compute a more complex aggregated quantitiy: the median. This computation hinges on the ability to compute the number of nodes. We propose such a technique but it only gives estimation and hence the median function is only estimated.

We consider this result to be significant because (i) the problem of computing aggregated quantities is common in wireless sensor networks which is an area of increasing importance and (ii) the techniques that we use depend on the availability of prioritized MAC protocols that support a very large range of priority levels; such protocols have recently been proposed [2], implemented and tested [3].

The remainder of this paper is structured as follows. Section 2 presents the system model and properties of the MAC protocol that we use. Section 3 shows how to compute the aggregated quantities. Section 4 shows how to estimate the number of proposed elements. Section 5 evaluates the algorithm for computing the number of elements. Section 6 discusses related work and this work. Section 7 gives conclusions.

2. System model

Consider a computer system comprised of m computing nodes that communicate over a wireless channel. Nodes do not have a shared memory; all data variables are local to each node. A computer node can make a wireless broadcast. This broadcast can be an unmodulated carrier wave or a message of data bits. We assume that all messages sent by nodes are related to computations of aggregate quantities. A node can transmit an empty message; that is, a message with no data. Every signal transmitted (unmodulated carriers or modulated data bits) is received by all computer nodes. This implies that there are no hidden stations and the network provides reliable broadcast.

Every node has an implementation of a MAC protocol. This MAC protocol is prioritized and collision-free. The fact that it is prioritized means that the MAC protocol assures that of all nodes that request to transmit at a moment, the one with the highest priority will transmit its data bits. The fact that it is collision-free implies that if priorities are unique then there is at most one node which transmits the data bits.

We assume that this MAC protocol is a dominance protocol. It operates as follows. The priority is encoded as a binary number with "0":s and "1":s. We say that a "0" is a dominant bit and a "1" is a recessive bit. We say that a low

number represents a high priority. This is similar to the CAN bus [4]. Computer nodes agree on an instant when the tournament starts. Then nodes transmit the priority bits starting with the most significant bit. Priority bits are modulated using a variation of On-Off keying. A node sends an unmodulated carrier wave if it had a dominant bit and it sends nothing if it had a recessive bit. In the beginning of the tournament, all nodes have the potential to win but if it was recessive at a bit and perceived a dominant bit then it withdraws from the tournament and it cannot win. When a node has won the tournament, then it clearly knows the priority of the winner. If a node has lost the tournament then it continues to listen in order to know the priority of the winner.

The operating system exposes three system calls for interacting with other nodes. The `send` system call takes two parameters, one describing the priority of the message and one describing the data bits to be transmitted. If `send` loses the tournament then it waits until a new tournament starts. The program making this system call blocks until a message is successfully transmitted. The function `send_empty` takes only one parameters and it is a priority. Interestingly, `send_empty` does not take any parameter describing the data. The system call `send_empty` works like the function `send` but if it wins it does not send anything. In addition, when the tournament is over (regardless of whether the node wins or loses), the function `send_empty` gives the control back to the application and the function `send_empty` returns the priority of the winner. There is also a function `just_listen` which works like `send_empty` but it loses even before the first bit, so `just_listen` will only return the priority of the winner.

We assume that a computer node proposes a value. This value may be a sensor reading such as a temperature. Computer node N_i proposes the value v_i. The range of the value v_i is known; it is [*MINV*..*MAXV*], we assume $0 \leq MINV$. For example it could be a 12 bit non-negative integer. Then the range is [0..4095]. All v_i have the same range for all proposed values. We assume that computer nodes do not know m.

We consider the problem of computing $f(v_1,v_2,\ldots,v_n)$ efficiently. We say that f is an *aggregated* quantity. We assume that there is one or many nodes that initiate the computation of f. When a node i has heard from one of these nodes that initiate the computation then node i proposes its value v_i. Every node has the potential to initiate a computation.

3. Computing aggregated quantities

We will first compute two simple quantities exactly in Section 3.1 and Section 3.2 and then, Section 3.3 shows how to compute a more complex quantity.

3.1. Computing the minimum value

Consider the case where the quantity that we want to compute $f(v_1,v_2,\ldots,v_m)$ is $min(v_1,v_2,\ldots,v_m)$. This can be performed as follows:

Algorithm 1. Calculating Min

```
When a node requests that min should be computed:
   Broadcast a message INITIATE_MIN
end
When a message INITIATE_MIN is received:
   Node i calculates value v_i that it proposes.
   minv := calcmin( vi )
end
subroutine calcmin( vi )
   return send_empty( priority = vi )
end
```

3.2. Computing the maximum value

Let us consider the computation of $f(v_1,v_2,\ldots,v_m)$ is $max(v_1,v_2,\ldots,v_m)$. This can be performed as follows:

Algorithm 2. Calculating Max

```
When a node requests that max should be computed:
   Broadcast a message INITIATE_MAX
end
When a message INITIATE_MAX is received:
   Node i calculates value v_i that it proposes.
   maxv := calcmax( vi )
end
subroutine calcmax( vi )
   return MAXV-send_empty( priority = MAXV - vi )
end
```

3.3. Computing the median value

We now consider the case where the function that we want to compute is the median of $v_1,v_2,\ldots,v_m$. We will find it convenient to introduce the notation $V_{less}(q)$ and $V_{greater}(q)$ as:

$$V_{less}(q) = \{v_j : v_j \leq q\} \quad (1)$$

$$V_{greater}(q) = \{v_j : v_j \geq q\} \quad (2)$$

With these definitions our goal is to find q such that $\|V_{greater}(q)|-|V_{less}(q)\|$ is minimized. We assume the existence of the function `get_n_elements_in( LB, UB, active)`. It will be described in Section 4 and it returns the number of computer nodes that proposed a value which is greater than or equal to LB and less than or equal to UB.

Algorithm 3. Calculating median value

```
When a node requests that median should be
  computed:
   Broadcast a message INITIATE_MEDIAN
end
When a message INITIATE_MEDIAN is received:
   Node i calculates value vi that it proposes.
   median := calcmedianvalues( vi )
end
subroutine calcmedianvalue( vi )
   LB := MINV
   UB := MAXV
   for j:=1..to log2(MAXV-MINV) do
      mid := ( LB + UB ) / 2
      active   :=vi<=mid
      nVless   :=get_n_elements_in(LB,mid,active)
      active   :=vi>=mid
      nVgreater:=get_n_elements_in(mid,UB,active)
      if nVless<=nVgreater then
        LB := mid
      else
        UB := mid
      end if
   endfor
   return mid
end
```

4. Computing the number of proposed elements

Computing the number of proposed nodes is equivalent to computing the number of nodes. However, computing this is non-trivial. Consider a node i that proposes a value v_i. All nodes will receive a value R from `send_empty`. If $R = v_i$ then node i cannot know if it is the only node (and hence $m = 1$) or there are many other nodes with $v_i = R$ as well. In fact, with the use of our MAC protocol this is impossible to achieve for an algorithm where all nodes makes a single call to `send_empty` at the same time. Based on this impossibility, we will focus on algorithms that do not find the exact value of m, but try to find an estimate of m. The intuition is that each computer node generates a random number and if there is a large number of nodes then the minimum random number is very small. We repeat this k times. Hence a large value of k gives a good accuracy of the estimate whereas a low value of k has low time-complexity. We think k=5 is a reasonable compromise (which will be discussed later). Algorithm 4 describes this.

Algorithm 4. Calculating nelements

```
When a node requests that number of elements
  should be computed:
   Broadcast a message INITIATE_NELEMENTS
When a message INITIATE_NELEMENTS is received:
   nnodes :=get_n_elements_in(MINV, MAXV, TRUE)
end

subroutine get_n_elements_in( LB, UB, active)
   for q:=1 to k do
     if active then
       R[q] := send_empty(priority = random(LB,UB) )
     else
       R[q] := just_listen
     end if
   end
   return ML_estimation( R[1],…,R[k], LB, UB )
end
subroutine ML_estimation( R, LB, UB )
   for q:=1 to k do
     u[q] := (UB-R[q])/(UB-LB)
   endfor
   loginvsum := 0
   for q := 1 to k do
     loginvsum := logsinvsum + ln( 1/u[q] )
   endfor
   return k/loginvsum
end
```

In Algorithm 4, we conveniently ignore the possibility of an interval with no nodes. We can understand the function `ML_estimation` by considering the following analysis. Let A_j denote the event that there were j nodes. Let $B(R^k)$ denote the event that the minimum of the proposed values is R^l when we generated random numbers the l:th time.
Let $B(R)=B(R^1)\cap B(R^2)\cap\ldots B(R^k)$. Let A_j denote the event that there are j nodes. When we have the minimum of the proposed values (in Algorithm 4) we wish to compute $P(A_j|B(R))$ for all values of j and select the value of j that maximizes

$$P(A_j|B(R)) \tag{3}$$

We will do so now. We know from Bayes´s formula that:

$$P(A_j|B(R)) = \frac{P(B(R)|A_j)\times P(A_j)}{\sum_{i=1}^{\infty}(P(B(R)|A_i)\times P(A_i))} \tag{4}$$

Let us assume that:

$$\forall j : P(A_1) = P(A_j) \tag{5}$$

Applying (5) in (4) gives us:

$$P(A_j|B(R)) = \frac{P(B(R)|A_j)}{\sum_{i=1}^{\infty}P(B(R)|A_i)} \tag{6}$$

Let us now compute $P(B(R)|A_j)$. We know that:

$$P(B(R)|A_i) = P(B(R^1)|A_i)\times\ldots\times P(B(R^k)|A_i) \tag{6b}$$

We obtain.

$$P(B(R^k)|A_i) = \frac{d}{dR}CDF(B(R^k)|A_i) \tag{7}$$

where CDF is the probability that the minimum is less than or equal to R. We compute it as follows. The probability that a random number is greater than or equal to R is

$$P(random\quad number \geq R) = \frac{MAXV - R}{MAXV - MINV} \tag{8}$$

The probability that the minimum of the i randomly generated numbers is greater than or equal to R is

$$P(\min imum\quad number \geq R) = \left(\frac{MAXV - R}{MAXV - MINV}\right)^i \tag{9}$$

Hence, we obtain:

$$CDF(B(R^k)|A_i) = 1 - \left(\frac{MAXV - R}{MAXV - MINV}\right)^i \tag{10}$$

Combining (10) with (7) gives us:

$$P(B(R^k)|A_i) = i\times\left(\frac{MAXV - R}{MAXV - MINV}\right)^{i-1} \tag{11}$$

Inserting (11) in (6) gives us:

$$P(A_j|B(R^k)) = \frac{j\times\left(\frac{MAXV - R}{MAXV - MINV}\right)^{j-1}}{\sum_{i=1}^{\infty}\left(i\times\left(\frac{MAXV - R}{MAXV - MINV}\right)^{i-1}\right)} \tag{12}$$

We wish to find the j that maximizes $P(A_j|B(R))$. We observe that this depends only on the numerator. Hence, we want to find the value of $j_{solution}$ that maximizes:

$$\prod_{q=1}^{k}\left(j_{solution}\times\left(\frac{MAXV - R^q}{MAXV - MINV}\right)^{j_{solution}-1}\right) \tag{13}$$

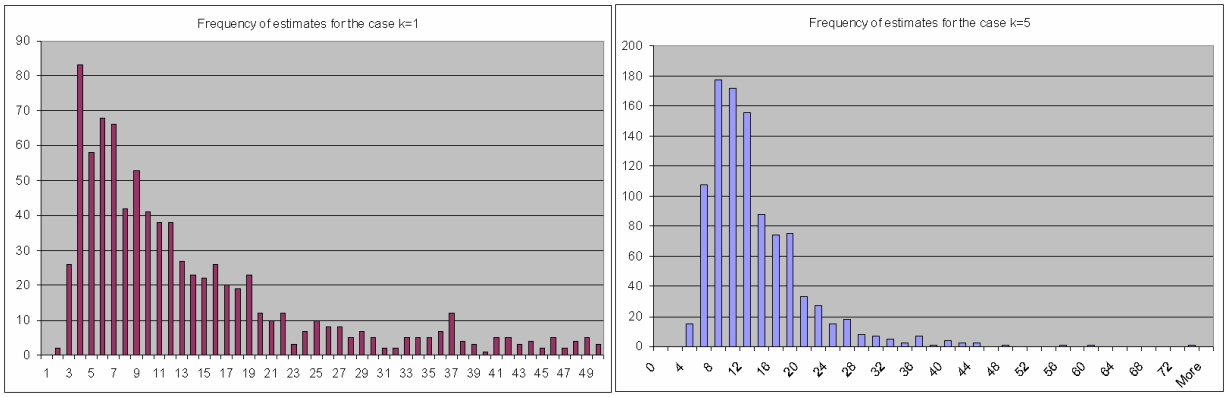

Figure 1. The frequency of the estimates for different values of *k*.

We can simplify (13) further. Let us use the notation:

$$u_q = \frac{MAXV - R^q}{MAXV - MINV} \tag{14}$$

and rewrite (13) we obtain that we want to maximize:

$$\prod_{q=1}^{k}\left(j_{solution} \times u_q^{\,j_{solution}-1}\right) \tag{15}$$

Observe that maximizing (15) is equivalent to maximizing the natural logarithm of (15). We know that the logarithm of a product is the sum of the logarithm of the factors. Hence, we want to maximize:

$$\sum_{q=1}^{k}\ln\left(j_{solution} \times u_q^{\,j_{solution}-1}\right) \tag{16}$$

We can rewrite (16) into the problem we want to maximize:

$$\sum_{q=1}^{k}\left(\ln j_{solution} + (j_{solution}) \times \ln u_q\right) \tag{17}$$

We have that the first derivative of (17) with respect to $j_{solution}$ is:

$$\sum_{q=1}^{k}\left(\frac{1}{j_{solution}} + \ln u_q\right) \tag{18}$$

And the second derivative of (17) with respect to $j_{solution}$ is:

$$-\sum_{q=1}^{k}\frac{1}{j_{solution}^{\,2}} \tag{19}$$

We can see from (18) and (19) that finding the $j_{solution}$ such that (18) is equal to 0 gives us the maximum likelihood estimate. Hence, we should select $j_{solution}$ such that:

$$\sum_{q=1}^{k}\left(\frac{1}{j_{solution}} + \ln u_q\right) = 0 \tag{20}$$

We can rewrite (20) to:

$$\frac{k}{j_{solution}} = -\sum_{q=1}^{k}\ln u_q \tag{21}$$

Rewriting yields:

$$\frac{1}{j_{solution}} = \frac{\sum_{q=1}^{k}\ln\frac{1}{u_q}}{k} \tag{22}$$

Further rewriting yields:

$$j_{solution} = \frac{k}{\sum_{q=1}^{k}\ln\frac{1}{u_q}} \tag{23}$$

This is a simple way to compute our estimate and we can see that `ML_estimation` in Algorithm 5 is based on this equation. We think it is simple enough to be used in a mote, although motes have very low processor speed.

5. Performance evaluation of nodes estimation

We have already mentioned that the calculation of the complex function median depends on the estimation of the number of nodes that propose a value. Hence, it is important that this estimation is accurate. For this purpose, we evaluate the accuracy using simulation experiments. Figure 1 shows the experimental results.

We ran 1000 experiments. For every experiment, 10 nodes generate random numbers and estimate the number of nodes. The estimation was made using (23). We can see that using five random numbers gives a significant improvement in the accuracy of the estimation as compared to one random number.

6. Related work and Discussion

6.1. Related work

A prioritized MAC protocol is useful to schedule real-time traffic [2, 3] and it can support data dissemination when topology is unknown [5]. In this paper we have discussed how to efficiently compute aggregated quantities using a prioritized MAC protocol.

Distributed calculations have been performed in previous research. It has been observed that nodes often [6, 7] detect an event and then needs to spread the knowledge of this event to its neighbours. This is called [6] one-to-*k* communication because only *k* neighbours need to receive the message. After that, the neighbour nodes perform local computations and reports back to the node that made the request for 1-to-*k* communication. This reporting back is called *k*-to-1 communication. Algorithms for both 1-to-*k* and *k*-to-1 communication are shown to be faster than naïve algorithm but unfortunately, the time-complexity increases as *k* increases. Our algorithms computes a function *f* and takes parameters from different nodes; this is similar to the average calculations in [8] . However our algorithms are different from [6, 7]; our algorithms have a time-complexity that does not depend on the number of nodes. We think our new algorithms are also useful building blocks for leader election and clock synchronization.

In this paper, nodes are permitted to use duplicated priorities, so any message transmitted after the tournament could collide and, for this reason, we use a send_empty primitive. However, it would be easy to code the priority in such a way that it would be unique by concatenating the node identifier to the priority. In this way, nodes could send a valid data message after winning the tournament. This is useful to because we may want to know not only the maximum value (for example the maximum temperature) but also other related values (for example the position of the node that detected the maximum temperature).

One way to use these algorithms is to encapsulate them in a query processor for database queries. Query processors for sensor networks have been studied in previous work [9, 10] but they are different in that they operate in multhop environment, do not compute aggregated quantities as efficiently as we do. They assume one single sink node and that the other nodes should report an aggregated quantity to this sink node. The sink node floods its interest in the data it wants into the network and this also makes nodes to discover the topology. When a node has new data it, broadcasts this data; other nodes hear it and it is routed and combined so that the sink node receives the aggregated. These works exploit the broadcast characteristics of the wireless medium (like we do) but they do not make any assumption on the MAC protocol (and hence they do not take advantage of the MAC protocol). One important aspect of these protocols is to create a spanning tree. It is known that computing an optimal spanning tree for the case when only a subset of nodes can generate data is equivalent to finding a Steiner-tree, a problem known to be NP-hard (the decision problem is NP-complete, see page 208 in [11]). For this reason, approximation algorithms have been proposed [12, 13]. However, in the average case, very simple randomized algorithms perform well [14]. Since a node will forward its data to the sink using a path which is not necessarily the shortest path to the sink, these protocols cause an extra delay. Hence, there is a trade-off between delay and energy-efficiency. To make this trade-off, a framework based on feedback was developed [15] for computing aggregated quantities. Techniques to aggregate data in the network such that the user at the base station can detect whether one node gives faked data has been addressed as well [16].

It has been observed that computing the median is especially difficult in multihop networks because combining two medians from different subnetworks is requires large amount of memory. Researchers in [17] observed that it is necessary for packets forwarded to be bigger and bigger the closer they get to the base station. Several algorithms for computing the exact median in O(*m*) time complexity are available (the earliest one is [18]). Our algorithm is faster; it has the time complexity O(log (MAXV-MINV)) but at the expensive of the accuracy of the result.

Computing averages has been done under the assumption that an adversary generates faults [19]. Unfortunately, it has a time-complexity which is larger than our algorithm and also larger than the algorithm proposed by [18] .

6.2. Practical issues

It is beyond the scope of this paper to describe all the details of the MAC protocol (see [2, 3] for details); it is important to observe however that the MAC protocol has the following properties. First, a priority bit has a duration adapted to time-of-flight, Rx/Tx switching time and time to detect a carrier and the duration of this bit can be quite large whereas a bit in the data packet has normal duration (for example on the CC2420 transceiver with a speed of 250kbps, a bit takes 4us). Hence, unlike CAN, in our protocol, the bit rate of the data transmission has the potential to be high even on long distances. Second, before the tournament in the protocol starts, the tournament waits for a long time of silence and synchronizes. This implies that even if nodes start the execution of the algorithms at slightly different times then the priority bits will be compared properly. This scheme only works if the different in time when message transmit messages "simultaneously" is not too big. We believe this assumption can easily be true however, by letting the algorithm start when it receives a message from a master node ordering the other nodes to start the execution of the algorithm.

So far we have assumed that all messages transmitted deal with aggregated quantities and we have assumed that there is only one type of aggregated quantity that we want to

compute. This can be solved easily. We can subdivide the priority field into two subfield. The most significant bits are called service identifier and the least significant bits are called data bits. For example, we have 10 priority bits; the 4 most significant bits could be the service identifiers and the remaining 6 bits are priority bits. The MAC protocol runs the tournament base on all 10 bits. If the 4 services bits are 0000 then the following 6 bits denotes the priority of a normal message and these 6 bits number represent a unique priority and is normal payload and it is collision free. If the 4 bits are 0001 it means that the 6 remaining contains data that should be used to compute the maximum temperature. An application can make a function call `send_empty` (0001, 20) which proposes the value 20 and returns the maximum temperature.

7. Conclusions

We have shown how to use a prioritized protocol to compute aggregated quantities efficiently. The computational complexity for min and max is O(log2(MAXV-MINV)), that is they do not depend on the number of nodes. Our estimation of the median can be computed efficiently as well, its time complexity is $O(k*[\log2(MAXV-MINV)]^2)$.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393-422, 2002.

[2] B. Andersson and E. Tovar, "Static-Priority Scheduling of Sporadic Messages on a Wireless Channel," presented at International Conference on Principles of Distributed Systems (OPODIS´05), Pisa, Italy, 2005.

[3] N. Pereira, B. Andersson, and E. Tovar, "Implementation of a Dominance Protocol for Wireless Medium Access," presented at IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Sydney, Australia, 2006.

[4] Bosch, "CAN Specification, ver. 2.0, Robert Bosch GmbH, Stuttgart," 1991.

[5] B. Andersson, N. Pereira, and E. Tovar, "Disseminating Data Using Broadcast when Topology is Unknown," presented at Proceedings of the 26th IEEE Real-Time Systems Symposium, Work-in-Progress Session, Miami Beach, Florida, 2005.

[6] R. Zheng and L. Sha, "MAC Layer Support for Group Communication in Wireless Sensor Networks," Department of Computer Science, University of Houston UH-CS-05-14, July 21 2005.

[7] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks," presented at Third European Workshop on Sensor Networks, Zurich, Switzerland, 2006.

[8] D. S. Scherber and H. C. Papadopoulos, "Distributed computation of averages over ad-hoc networks," *IEEE J. Select. Areas Commun*, vol. 23, pp. 776-787, 2005.

[9] Y. Yao and J. E. Gehrke, "Query Processing in Sensor Networks," presented at Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003), Asilomar, California, 2003.

[10] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," presented at Proceedings of OSDI, Boston, MA., 2002.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability A guide to the Theory of NP-Completeness* New York: W. H. Freeman and Company, 1979.

[12] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," presented at 22nd International Conference on Distributed Computing Systems, 2002.

[13] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," presented at Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, Austria, 2002.

[14] M. Enachescu, A. Goel, R. Govindan, and R. Motwani, "Scale Free Aggregation in Sensor Networks," presented at First International Workshop on Algorithmic Aspects of Wireless Sensor Networks, 2004.

[15] T. Abdelzaher, T. He, and J. Stankovic, "Feedback Control of Data Aggregation in Sensor Networks," presented at 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas, 2004.

[16] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," presented at ACM SenSys (Conference on Embedded Networked Sensor Systems)], 2003.

[17] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks. SenSys 2004: 239-249," presented at SenSys, Baltimore, Maryland, USA, 2004.

[18] M. Blum, R. W. Floyd, V. Pratt, R. Rivest, and R. Tarjan, "Time bounds for selection," *J. Cornput. System Sci*, vol. 7, pp. 448-461., 1973.

[19] M. Kutylwski and D. Letkiewicz, "Computing Average Value in ad hoc Networks," presented at MFCS 2003 28th International Symposium on Mathematical Foundations of Computer Science, Bratislava, Slovak Republic, Europe, 2003.

Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks

Anis KOUBAA[1], Mário ALVES[1], Bilel NEFZI[2], Ye-Qiong SONG[2]

[1] IPP-HURRAY! Research Group, Polytechnic Institute of Porto
Rua Dr. Antonio Bernardino de Almeida, 431, 4200-072 Porto, PORTUGAL

[2] LORIA-TRIO, 615 rue du jardin botanique 54600 Villers Les Nancy FRANCE

akoubaa@dei.isep.ipp.pt, mjf@isep.ipp.pt, bilel.nefzi@loria.fr, song@loria.fr

Abstract

In beacon-enabled mode, IEEE 802.15.4 is ruled by the slotted CSMA/CA Medium Access Control (MAC) protocol. The standard slotted CSMA/CA mechanism does not provide any means of differentiated services to improve the quality of service for time-critical events (such as alarms, time slot reservation, PAN management messages etc.). In this paper, we present and discuss practical service differentiation mechanisms to improve the performance of slotted CSMA/CA for time-critical events, with only minor add-ons to the protocol. The contribution of our proposal is more practical than theoretical since our initial requirement is to leave the original algorithm of the slotted CSMA/CA unchanged, but rather tuning its parameters adequately according to the criticality of the messages. We present a simulation study based on an accurate model of the IEEE 802.15.4 MAC protocol, to evaluate the differentiated service strategies. Four scenarios with different settings of the slotted CSMA/CA parameters are defined. Each scenario is evaluated for FIFO and Priority Queuing. The impact of the hidden-node problem is also analyzed, and a solution to mitigate it is proposed.

1. Introduction

Providing Quality of Service (QoS) support in Wireless Sensor Networks (WSNs) for improving their timing and reliability performance under severe energy constraints has attracted recent research works [1-3]. The standardization efforts of the IEEE Task Group 15.4 have contributed to solve this problem by the definition of the IEEE 802.15.4 protocol for *Low-Rate*, *Low-Power* Wireless Personal Area Networks (WPANs) [4]. In fact, this protocol shows great potential for flexibly fitting different requirements of WSN applications by adequately setting its parameters (low duty cycles, guaranteed time slots (GTS)). In beacon-enabled mode, the IEEE 802.15.4 protocol uses slotted CSMA/CA as a Medium Access Protocol (MAC). Even though the IEEE 802.15.4 protocol provides the GTS allocation mechanism for real-time flows, the allocation must be preceded by an allocation request message. However, with its original specification, the slotted CSMA/CA does not provide any QoS support for such time-sensitive events, including GTS allocation requests, alarms, PAN management commands, etc., which may result in unfairness and degradation of the network performance, particularly in high load conditions.

Related work. The improvement of CSMA/CA MAC mechanisms has drawn many research efforts. Particularly for the case of the IEEE 802.15.4 protocol, some recent research works have contributed to enhance the slotted CSMA/CA mechanism for achieving reduced (soft) delay guarantees and better reliability of time-critical events, as described next.

In [5], the authors modified the slotted CSMA/CA algorithm to enable fast delivery of high priority frames in emergency situations, using a priority toning strategy. Nodes that have high priority frames to be transmitted must send a tone signal just before the beacon transmission. If the tone signal is detected by the PAN Coordinator, an emergency notification is conveyed in the beacon frame, which alerts other nodes with no urgent messages to defer their transmissions by some amount of time, in order to privilege high priority frame transmissions at the beginning of the contention access period. In [6], the authors extend the previous schemes by allowing high priority frames to perform only one *Clear Channel Assessment* (CCA) operation instead of two, using a frame tailoring strategy, which aims to avoid collisions between data frames and acknowledgment frames when only one CCA is performed. These solutions seem to improve the responsiveness of high priority frames in IEEE 802.15.4 slotted CSMA/CA, but require a non-negligible change to the IEEE 802.15.4 MAC protocol to support the priority toning and frame tailoring strategies, thus turning them non-compatible with the standard.

In this paper, we investigate other alternatives for improving slotted CSMA/CA without forcing fundamental changes to the MAC protocol. We particularly aim to assess different parameter settings of the protocol with some basic queuing strategies (FIFO and Priority Queuing) for each traffic priority. Note that in [5, 6], the toning mechanism imposes some changes to the hardware (using a tone signal transmitter) and also to the protocol itself, due to the frame tailoring strategy.

The motivation for proposing differentiated QoS support with only minor add-ons to the slotted CSMA/CA mechanism is to ensure backward compatibility with the standard. Also, we would like to assess if such a simple approach is sufficient to satisfy the requirements of time-critical messages. This proposal can be easily adopted in the IEEE 802.15.4b extension [7] of the standard.

The rest of the paper is organized as follows. Section 2 highlights the IEEE 802.15.4 features and its slotted CSMA/CA mechanism. Section 3 presents the proposed differentiation service strategies. Section 4 presents the simulation study and performance evaluation results. Section 5 concludes the paper.

2. IEEE 802.15.4 Slotted CSMA/CA MAC

In beacon-enabled mode, beacon frames are periodically sent by a central device, referred to as *PAN coordinator*, to identify its PAN and synchronize nodes that are associated with it. The PAN coordinator defines a superframe structure characterized by a *Beacon Interval* (*BI*) specifying the time between two consecutive beacons, and a *Superframe Duration* (SD) corresponding to the active period, defined as:

$$\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \\ for\ & 0 \le SO \le BO \le 14 \end{aligned} \quad (1)$$

BO and *SO* are called *Beacon Order* and *Superframe Order*, respectively. The Beacon Interval may optionally include an inactive period (for $SO < BO$), in which all nodes may enter into a sleep mode, thus saving energy. More details can be found in [4].

By default, nodes compete for medium access using slotted CSMA/CA during the *Contention Access Period* (CAP). The

IEEE 802.15.4 protocol also provides a *Contention-Free Period* (CFP) within the superframe, in which a node may request the PAN coordinator to allocate Guaranteed Time Slots (GTS). In this paper, we consider the physical layer operating in the 2.4 GHz frequency band and with a 250 kbps data rate.

The slotted CSMA/CA algorithm is based on a basic time unit called *Backoff Period* (BP), which is equal to $aUnitBackoffPeriod = 80$ bits (0.32 ms). The slotted CSMA/CA backoff algorithm mainly depends on three variables: **(1)** the *Backoff Exponent* (*BE*) enables the computation of the backoff delay, **(2)** the *Contention Window* (*CW*) represents the number of BPs during which the channel must be sensed idle before channel access, **(3)** the *Number of Backoffs* (*NB*) represents the number of times the CSMA/CA algorithm was required to backoff while attempting to access the channel. Fig. 1 presents the slotted CSMA/CA algorithm [4].

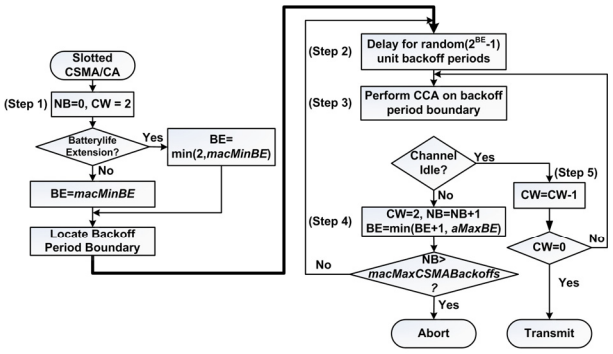

Fig. 1. The slotted CSMA/CA algorithm

First, the number of backoffs and the contention window are initialized ($NB = 0$ and $CW = CW_{init} = 2$) (Step 1). The backoff exponent is also initialized to $BE = 2$ or $BE = \min(2, macMinBE)$, depending on the value of the *Battery Life Extension* MAC attribute. *macMinBE* is a constant, which is by default equal to 3. Then, the algorithm starts counting down a random number of BPs uniformly generated within $[0, 2^{BE}-1]$ (Step 2). The count down must start at the boundary of a BP. When the timer expires, the algorithm then performs one CCA operation at the BP boundary to assess channel activity (Step 3). If the channel is busy (Step 4), *CW* is re-initialized to $CW_{init} = 2$, *NB* and *BE* are incremented. *BE* must not exceed *aMaxBE* (default value fixed to 5). Incrementing *BE* increases the probability for having greater backoff delays. If the maximum number of backoffs ($NB = macMaxCSMABackoffs = 5$) is reached, the algorithm reports a failure to the higher layer; otherwise, it goes back to (Step 2) and the backoff operation is restarted. The protocol allows $aMaxFrameRetries = 3$ after each failure. If the channel is sensed as idle, *CW* is decremented (Step 5). The CCA is repeated if $CW \neq 0$. This ensures performing two CCA operations to prevent potential collisions of acknowledgement frames. If the channel is again sensed as idle, the node attempts to transmit, provided that the remaining BPs in the current CAP are sufficient to transmit the frame and the subsequent acknowledgement. If not, the CCAs and the frame transmission are both deferred to the next superframe. This is referred to as *CCA deference*.

3. Service Differentiation Strategies for Slotted CSMA/CA

Observe that the behavior of slotted CSMA/CA is affected by four *initialization* parameters, which are: (1) the minimum backoff exponent (*macMinBE*), (2) the maximum backoff exponent (*aMaxBE*), (3) the initial value of the *CW* (CW_{init}) and (4) the maximum number of backoffs (*macMaxCSMABackoffs*).

Changing the value of any of these parameters will have an impact on the performance. For instance, a performance evaluation study in [8] has shown that the average delay of broadcast frames increases with *macMinBE*, whereas the probability of success remains independent of *macMinBE* in large-scale WSNs. However, the probability of success increases for high *macMinBE* values, in small-scale WSNs. Based on those observations, we propose to offer differentiated services for time-critical messages. In this paper, our service differentiation mechanisms are particularly based on the *macMinBE*, *aMaxBE* and CW_{init} parameters.

Note that IEEE 802.15.4 defines two frame types: (1) *data traffic*, which typically represents sensory data broadcasted to the network (without using acknowledgments), (2) and *command traffic*, which comprises critical messages (such as alarm reports, PAN management messages and GTS requests) sent by sensor nodes to the PAN Coordinator. Due to their importance, command frames are sent using acknowledged transmissions to ensure the reliability of their transfer, and require a particular QoS support to be delivered to their destination in a bounded time interval. In this paper, we consider command frames as the *high priority service class* and data frames as the *low priority service class*.

The differentiated service strategies are presented in Fig. 2.

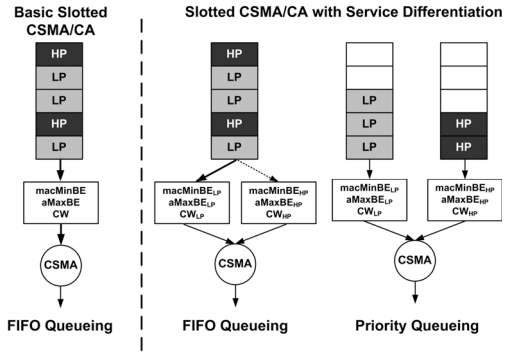

Fig. 2. Differentiated service strategies

The idea is simple. Instead of having the same CSMA/CA parameters for both traffic types, we assign each class its own attributes. We denote [$macMinBE_{HP}$, $aMaxBE_{HP}$] and CW_{HP} the backoff interval and the contention window initial values for high priority traffic related to command frames, and [$macMinBE_{LP}$, $aMaxBE_{LP}$] and CW_{LP} the initial values for low priority traffic related to data frames. While, the slotted CSMA/CA described in Section 2 remain unchanged, the adequate initial parameters that correspond to each service class must be applied.

In addition to the specification of different CSMA/CA parameters, Priority Queuing can be applied to reduce queuing delays of high priority traffic (Fig. 2). In this case, slotted CSMA/CA uses priority scheduling to select frames from queues, and then applies the adequate parameters corresponding to each service class. Note that if a low priority frame is selected, i.e. the high priority queue is empty, then the backoff process corresponding to this frame will not be preempted, if a high priority frame arrives during that service time, until this frame is sent, or rejected if the maximum number of backoff is reached.

The heuristics for adequately setting the CSMA/CA parameters are the following. Intuitively, a first differentiation consists in setting CW_{HP} smaller than CW_{LP}. It results that low priority traffic has to assess the channel to be idle for a longer time before transmission. A second differentiation is related to the backoff interval. Providing lower backoff delay values for high priority traffic by setting $macMinBE_{HP}$ lower than $macMinBE_{LP}$ would improve its responsiveness without degrading its throughput, as it has been observed in [8]. These intuitive heuristics are evaluated in the next section.

4. Performance Evaluation

4.1 Simulation Workload and scenarios

We present a simulation study based on an accurate model of IEEE 802.15.4 using OPNET simulator [9], to assess the impact of differentiated services in slotted CSMA/CA. We consider a WSN in a surface of 100 m x 100 m with one PAN coordinator, *BO* = *SO* = 3 and 100 identical nodes (randomly spread) generating *low priority* (*data*) *traffic* with Poisson distributed arrivals with the same mean arrival rate. The data frame size is fixed to 404 bits (300 bits of data payload + 104 bits of MAC header). These nodes also generate *high priority* (*command*) *traffic* with an inter-arrival time exponentially distributed with a mean value equal to 1 second. The command frame size is fixed to 304 bits (200 bits of data payload + 104 bits of MAC header). Frame size values are chosen as illustrative examples of short frame sizes. Command frames are sent from nodes to the PAN Coordinator using acknowledged transmissions. Data frames are simply broadcasted to the network. The maximum number of backoffs *macMaxCSMABackoffs* is equal to 4 and the maximum number of retries *aMaxFrameRetries* is by default equal to 3. The transmission power is equal to 0.1 mW.

The simulation study consists in varying the intensity of data traffic, while the command frames remain exponentially generated with the average of 1 frame/second in each node, and analyzing the performance of command frames in terms of average delay (*D*), probability of success (*S/Gapp*) and power efficiency. *S* denotes the throughput of command frames and *Gapp* denotes the offered load of command frames generated by the application layers of 100 nodes. In this study, *Gapp* is approximately equal in average to 31.5 kbps (= 100 * 304 bits per second), which represents 12.5% of the overall network capacity (250 kbps).

Note that there is a difference between *Gapp* and *Gmac*. The latter is defined as the offered load generated by the MAC layers due to the transmissions of original command frames and the retransmissions of their copies in case of non successful delivery. Hence, the power efficiency is reflected by the *Gmac* performance metric, i.e. fewer retransmissions (lower *Gmac*) results in a better power efficiency.

In this paper, the performance of data frames is also analyzed in terms of average delay and probability of success $\left(S_{data}/G_{mac}^{data}\right)$, which reflects the degree of reliability achieved by the network for successful transmissions of data frames. In case of data traffic, the probability of success is measured by the throughput of data frames S_{data} divided by the offered load of data frames generated by the MAC layers (G_{mac}^{data}). Since there is no retransmissions in case of a transmission failure of a data frame (unacknowledged transmissions), G_{mac}^{data} is at most equal to G_{app}^{data}, which is the data traffic generated by the application layer. This is because, at a given time, it may happen that some data frames are still waiting for service in the queue. Note that in our scenario with 100 nodes, we have verified that $G_{mac}^{data} = G_{app}^{data}$, for all network loads (*G*) considered in this simulation study (no buffer overflow for data frames). The network load (*G*) represents all command and data frames generated by the MAC layers of 100 nodes.

We consider four different scenarios, presented in Table 1. Each scenario is simulated with FIFO and Priority Queuing scheduling policies (refer to Fig. 2).

Table. 1. Simulation scenarios

Scenario	[macMinBE$_{HP}$, aMaxBE$_{HP}$]	[macMinBE$_{LP}$,aMaxBE$_{LP}$]	CW$_{HP}$	CW$_{LP}$
Sc1	[2,5]	[2,5]	2	2
Sc2	[2,5]	[2,5]	2	3
Sc3	[0,5]	[2,5]	2	2
Sc4	[0,5]	[2,5]	2	3

4.2 Case of a fully connected network (no hidden-node problem)

First, we consider a fully connected network, where all nodes hear each other.

Fig. 3 clearly shows the impact of the first differentiation scheme related to the initial contention window size on the success probability. As it was intuitively expected, setting CW_{LP} greater than CW_{HP} notably results in higher throughputs for high priority command frames, either for FIFO or Priority Queuing. The success probability remains satisfactory even in high load conditions for Sc2 and Sc4 (up to 80%). However, the effects of *macMinBE* and scheduling policies are negligible on *S*/*Gapp* since Sc1 and Sc3 have the same throughput (similarly to Sc2 and Sc4) for different *macMinBE* values. This confirms the result in [8].

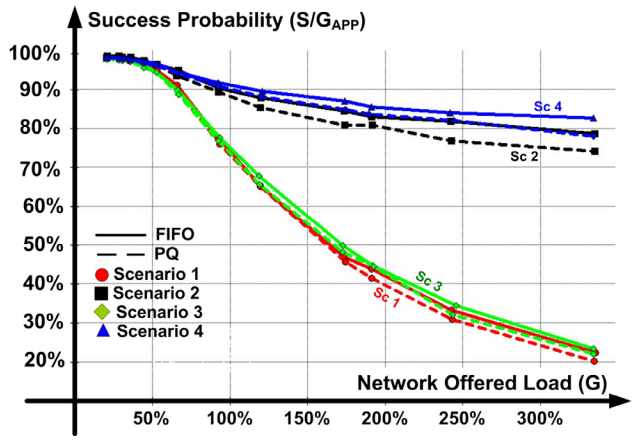

Fig. 3. Success probability of command frames without hidden nodes

Fig. 4 shows the average delays for all scenarios. Sc1 is only comparable to Sc3, whereas Sc2 is comparable to Sc4, due to the success probability results in Fig. 3 (it is not logical to compare delays for scenarios with different success probabilities).

Observe that lower *macMinBE* for high priority frame leads to lower average delays, since the backoff delays are reduced. The beauty of this result is that lower *macMinBE* does not degrade the throughput, as shown in Fig. 3. The advantage of using Priority Queuing in reducing average delays is also observable in Fig. 4.

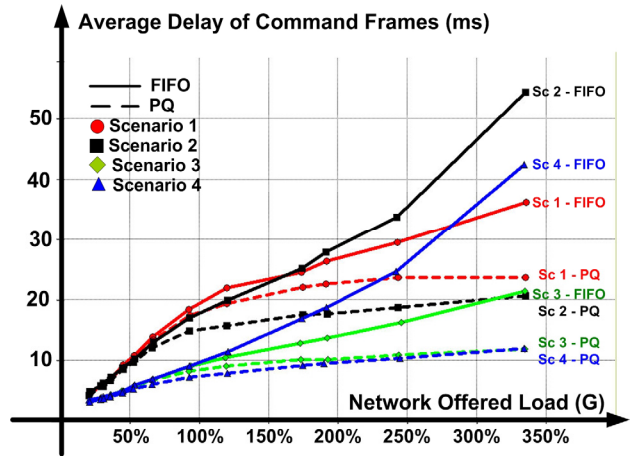

Fig. 4. Average delay of command frames (ms) without hidden nodes

As for power efficiency (Fig. 5), setting CW_{LP} greater than CW_{HP} clearly results in lower energy consumption, since fewer retransmissions are needed in Sc2 and Sc4. Priority Queuing seems also to be advantageous for improving energy efficiency.

The impact of *macMinBE* on *Gmac* depends on the values of CW_{LP} and CW_{HP}. If both are equal (Sc1 and Sc3), higher *macMinBEs* are more energy efficient. However, if $CW_{LP} < CW_{HP}$ (Sc2 and Sc4) lower *macMinBEs* are more energy efficient. This is because retransmissions are mostly due to collisions with data frames. Since Sc4 provides more differentiation to high priority frames than the other scenarios, it presents the best performance for all metrics.

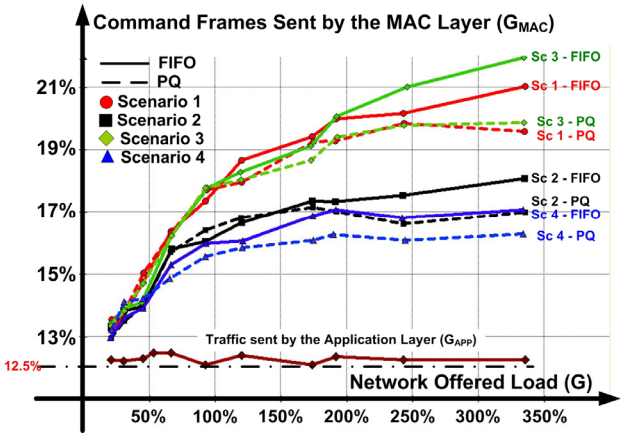

Fig. 5. Command traffic sent by the MAC layer without hidden nodes

As for the performance of low priority data frames, Figs. 6 and 7 present the success probability and the average delay, respectively. In Fig. 6, it is shown that setting CW_{LP} greater than CW_{HP} (Sc2 and Sc 4) results in relatively lower throughputs for low priority data frames, due to the privileges given to the high priority frames, as it can be observed in Fig. 3. However, the improvement of this differentiation scheme to the throughput of high priority command frames is more significant than the degradation of the throughout of low priority data frames, which further demonstrates the efficiency of this differentiation mechanism.

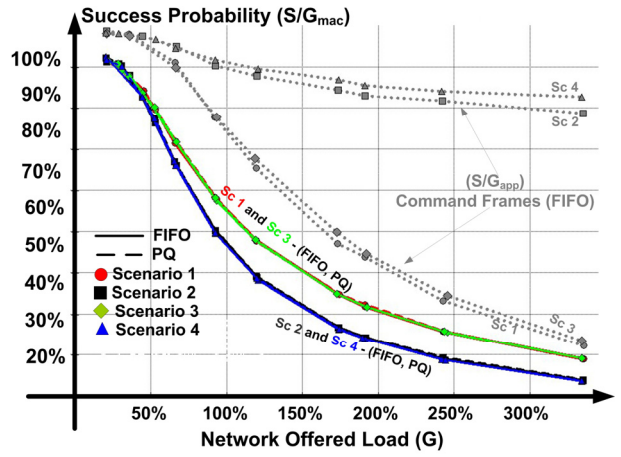

Fig. 6. Success probability of data frames without hidden nodes

In Fig. 7, observe that setting CW_{LP} greater than CW_{HP} results in greater average delays for data frames. This is because low priority data frames have a smaller probability to access the medium than high priority command frames when CW_{LP} increases, leading to return more often to the backoff process. This results in additional queuing and backoff delays (BE increases each time the channel is sensed busy) for data frames.

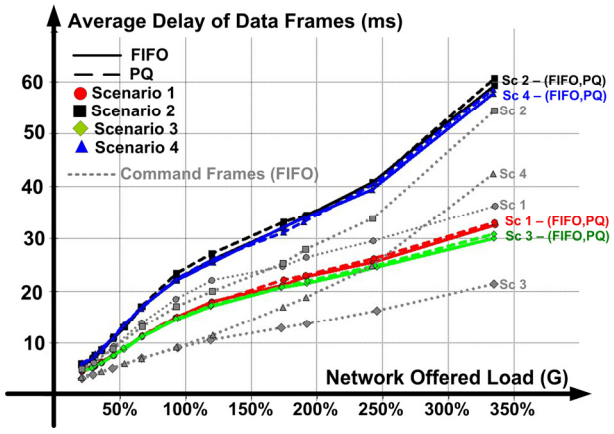

Fig. 7. Success probability of data frames without hidden nodes

Note that the Priority Queuing scheduling mechanism does not degrade the average delays of data frames even though they receive a low priority service. This is due to the fact that, in these simulation scenarios, command frames only use 12.5% (31.5 kbps) of the network capacity. The degradation would be more significant if command frames were generated at a higher rate. This behavior is typical for many WSNs, since command frames are likely to be generated with lower rate than data frames.

Another interesting observation is that the average delays of command frames are lower than those of data frames in all scenarios, except in Sc1 which does not provide any kind of differentiation. As a result, it is clearly shown that using one or both differentiation strategies (*CW* and/or *macMinBE*) always results in an improved performance for high priority frames.

4.3 Case of partially connected network (hidden-node problem)

We consider a partially connected network (we adjust the sensing sensitivity of the nodes to limit their communication range), to evaluate the impact of the hidden-node problem on the performance of slotted CSMA/CA with differentiated services. The sensing and receiving sensitivities are set such that the transmission range of each sensor node is limited to 32 m (command and data frames are sent with a transmission power equal to 0.1 mW). Beacon frames are sent by the PAN Coordinator at a transmission power equal to 1 mW, which is sufficient to reach all the nodes in the WSN. No routing protocol is used. Frames are simply broadcasted to the network (1) since most WSNs rely on broadcast transmissions and (2) we would like to provide results independent from any routing protocol.

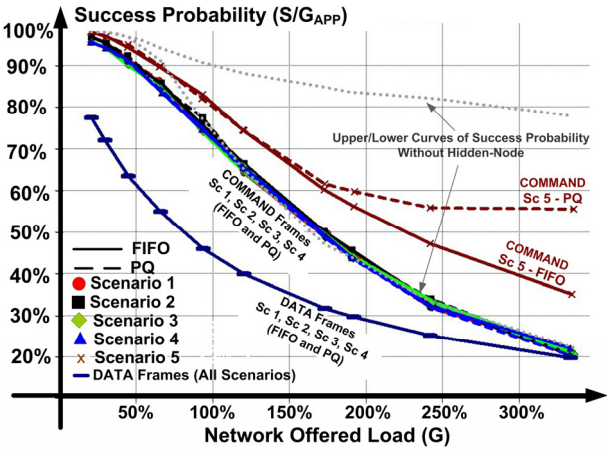

Fig. 8. Success probability of command/data frames with hidden nodes

It can be observed in Figs. 8, 9, 10 and 11 that the differentiated service strategies of the four scenarios defined in Table 1 have practically no impact on the performance metrics for both command and data frames, with an exception for the average delays. As shown in Fig. 9, lower *macMinBEs* slightly reduce the average delays of command frames. On the other hand, observe in Fig. 10 that greater CW_{LP} only results in a non significant increase of the average delays of low priority frames (difference around 1 ms). The success probabilities of command frames, as well as of data frames, remain closely insensitive to the differentiation service strategies in the four scenarios. In addition, The Priority Queuing scheduling policy has no impact on the improvement of the performance of high priority command frames.

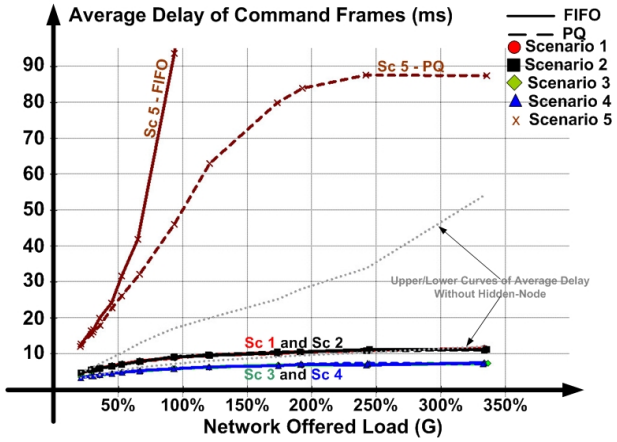

Fig. 9. Average delay (ms) of command frames with hidden nodes

These results clearly infer the severe impact of the hidden-node problem on the degradation of the performance of slotted CSMA/CA. Since nodes cannot hear each other, multiple hidden-node collisions occur independently of the differentiation schemes.

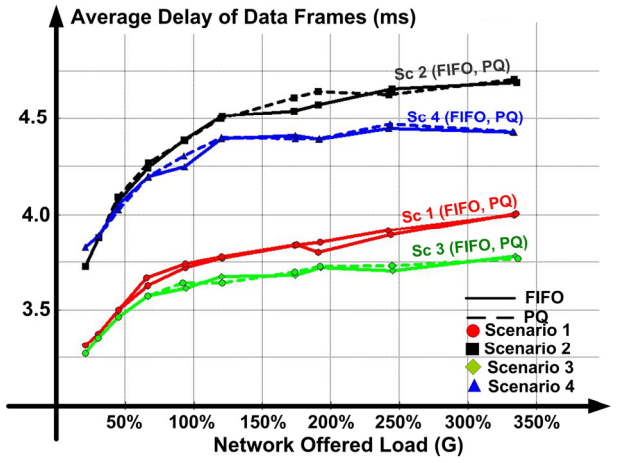

Fig. 10. Average delay (ms) of data frames with hidden nodes

The hidden-node impact is mainly a result of the small backoff interval duration. Note that with *aMaxBE* value equal to 5, the maximum backoff delay is equal to 31 BPs, which is not sufficient to avoid hidden-node collisions. One option to mitigate the hidden-node problem is to increase the backoff delay, such that competing nodes wait longer. Hence, other nodes would have more chance to successfully transmit their frames without facing hidden-node collisions. To illustrate this intuition, we propose the following additional scenario Sc5.

Table. 2. Hidden-node avoidance scenario

Scenario	[$macMinBE_{HP}$, $aMaxBE_{HP}$]	[$macMinBE_{LP}$,$aMaxBE_{LP}$]	CW_{HP}	CW_{LP}
Sc5	[4,6]	[7,8]	2	10

By increasing *macMinBE* and *aMaxBE* for both high priority and low priority traffics, the backoff delay will clearly increase for both traffic classes. Observe also that CW_{LP} is set to 10 and CW_{HP} is set to 2, to give more privileges to high priority frames.

It can be observed in Fig. 8 that the configuration of Sc5 noticeably improves the throughput of command frames, by reducing hidden-node collisions. With Priority Queuing in Sc5, the success probability reaches more that 55% even in high load conditions. However, reporting to Fig. 9, the average delays can be very large with FIFO scheduling, but are more steady using Priority Queuing (less than 90 ms).

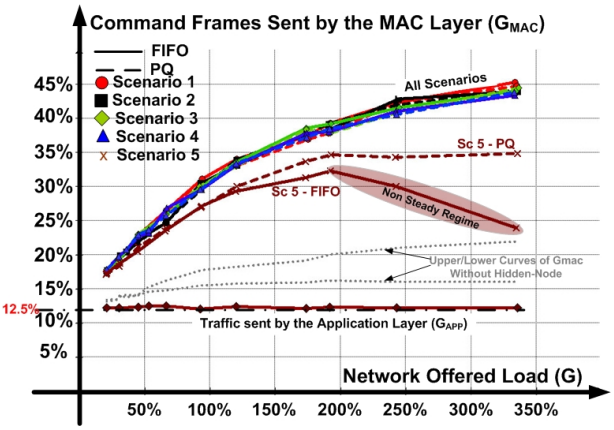

Fig. 11. Command traffic sent by the MAC layer with hidden nodes

Note that in Sc5 with FIFO, the network operates in a non steady regime (Fig. 11) in high load conditions, due to overloaded queues, which explains the expansion of average delays. The same behavior occurs for low priority data frames, both with FIFO and Priority Queuing. This is due to the blocking of high priority command frames by low priority data frames, which must wait for 10 CCA before transmission. However, with Priority Queuing, Sc5 is more energy efficient since fewer retransmissions than is other scenarios are performed.

5. Discussions

We have proposed a simple differentiated service scheme for slotted CSMA/CA in IEEE 802.15.4 to improve the performance of time-sensitive messages. It has been shown that tuning adequately the parameters of slotted CSMA/CA may result in an improved QoS for time-critical messages. This practical proposal can be easily adopted in the IEEE 802.15.4b extension of the standard since it only requires minor add-ons and ensures backward compatibility with the existing standard.

We have run the same simulation scenarios [10] using the implementation of the IEEE 802.15.4 protocol in the NS-2 simulator [11], for (1) comparative purposes, (2) the validation of our simulation results. The results obtained using NS-2 show a similar behavior to the results presented in this paper, thus confirming the validity of the approach. However, the values of the average delays observed in NS-2 results are greater than those obtained with our OPNET model. Also, NS-2 produces lower throughputs than those obtained with OPNET. To our understanding, this is mainly due to the amount of additional overheads introduced by the NS-2 simulator, since it imposes the use of a UDP (User Datagram Protocol) agent in each node for generating data, and also the generation of ARP (Address Resolution Protocol) frames. This is mainly because NS-2 was

originally developed for IP (Internet Protocol) networks and then extended for IEEE 802.11 wireless networks. According to our personal experience, we strongly believe that the current version of the NS-2 simulator is not accurate for the simulation of wireless sensor networks, even though existing modules can be reused in this context. Our OPNET model implements more accurately the IEEE 802.15.4 protocol without these unnecessary overheads, turning its results more reliable than those obtained with NS-2.

References

[1] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On Real-Time Capacity Limits of Multihop Wireless Sensor Networks," in Proceedings of the IEEE International Real-Time Systems Symposium, Lisbon, Portugal, 2004.

[2] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks," *Proceedings of the IEEE*, vol. 91, pp. 1002-1022, 2003.

[3] A. Koubâa, M. Alves, and E. Tovar, "IEEE 802.15.4: a Federating Communication Protocol for Time-Sensitive Wireless Sensor Networks," in *Technical Report TR-060201, to appear in Sensor Networks and Configurations: Fundamentals, Techniques, Platforms, and Experiments*, N. P. Mahalik, Ed. Germany: Springer-Verlag, 2006.

[4] IEEE-TG15.4, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE standard for Information Technology*, 2003.

[5] T. Kim, D. Lee, J. Ahn, and S. Choi, "Priority toning strategy for fast emergency notification in IEEE 802.15.4 LR-WPAN," in Proceedings of the 15th Joint Conference on Communications & Information (JCCI), April, 2005.

[6] T. Kim and S. Choi, "Priority-based delay mitigation for event-monitoring IEEE 802.15.4 LR-WPANs," *IEEE Communications Letters*, Nov. 2005.

[7] IEEE-TG15.4b, "http://grouper.ieee.org/groups/802/15/pub/TG4b.html."

[8] A. Koubâa, M. Alves, and E. Tovar, "A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks," in Proceedings of the 6th IEEE International Workshop on Factory Communication Systems (WFCS 2006), Torino (ITALY), 2006.

[9] OPNET, "OPNET Simulator, v 11, http://www.opnet.com."

[10] B. Nefzi, "Performance Evaluation and Improvement of the Slotted CSMA/CA MAC of the IEEE 802.15.4 Protocol ": Master Project, (LORIA, IPP-HURRAY and Sup'Com), 2006.

[11] NS-2, "NS-2 Simulator, http://www.isi.edu/nsnam/ns/."