SPARE MAC Enhanced: A Dynamic TDMA Protocol for Wireless Sensor Networks

Fabio Turati, Matteo Cesana
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Milan, Italy
{turati, cesana}@elet.polimi.it

Luca Campelli
CEFRIEL
Politecnico di Milano, Milan, Italy
Luca.Campelli@cefriel.it

***Abstract*—Wireless Sensor Networks may support heterogeneous applications ranging from classical environmental monitoring, to more demanding integrated surveillance systems based on multimedia traffic. To this extent, we argue that the specific MAC layer solutions must be flexible enough to provide differentiated services based on the context of the specific WSN.**

In this work, we leverage the SPARE MAC protocol for WSN based on a receiver-oriented TDMA protocol, and we propose adaptive techniques to dynamically adjust the resource (slots) assignment which are able to track the changes in the network status (traffic increase/decrease, contention increase/decrease). We comment on the trade-off between performance gain and increased signalling complexity, by resorting to numerical results derived through simulation.

I. Introduction

Wireless Sensor Networks (WSNs) [1] are multi-hop networks consisting of many nodes with sensing, processing, and radio communications capabilities, whose application fields may range from environmental monitoring, to the support of multimedia information transfer services. In general, network nodes in WSNs are battery powered, thus one critical aspect is saving energy to prolong network lifetime and avoid hard and expensive operations of battery replacement.

Wireless Multimedia Sensor Networks (WMSNs) [2][3] derive from WSNs and are emerging as a promising technology to support integrated surveillance systems [4]. Different from classical WSNs, since the support of video streams is required, the focus slightly switches from energy efficiency to Quality of Service (QoS) support, in terms of throughput and delay.

Traditionally, WSNs leverage Medium Access Control (MAC) schemes of the following two classes: Carrier Sense Multiple Access and/or Time Division Multiple Access. CSMA, which is used for example by the seminal SMAC [5] and all its subsequent enhancements, is usually chosen for its simplicity, as it doesn't require synchronization; however, the contention-based mechanism it is based on does not provide efficient support to QoS, being therefore better suited for applications where traffic is expected to be low; on the other hand, TDMA [6] is more difficult to manage, but its time-slotted structure allows wise (and efficient) assignment of the radio resources, as in TRAMA [7].

Static TDMA, however, does not fit WSNs requirements: indeed, the number of resources (slots) that must be assigned to each sensor, which plays a fundamental role in determining both the QoS and the energy consumptions, depends on the topology and on the amount of offered traffic, which, in turn, are dynamic. To this extent, the use of dynamic TDMA has already been investigated in the literature. In [8], the first slot of the frame is cyclically used by all sensors to announce and confirm transmitter-receiver pairs, thus the protocol results in a contention-free TDMA where all conflicting slot assignments are detected and resolved and scarce adaptation is possible. In [9], packets flow in only one direction, from sons to fathers of a tree, and sons can hand over unneeded slots to fathers, following the principle that unused slots are a waste. The authors describe an algorithm that avoids the creation of bottlenecks as slots are handed over. In [10] the frame length is dynamically changed to avoid having an excessive number of unassigned slots.

All the aforementioned work on TDMA/Dynamic TDMA tend to assign resources (slots) to the transmitting sensors. We focus here on a different approach to TDMA, that is, the receiver-oriented approach. The recently proposed SPARE MAC (Slot Periodic Assignment for REception) protocol [11], instead of assigning slots for *transmissions*, distributes resources for the *reception* of data. In this work, we leverage the SPARE MAC protocol and introduce techniques to make the resource assignment process dynamic and dependent on the context of the network. Namely, we propose algorithms and protocol adjustment to dynamically change the resource assignment policy depending on the current network context (interference, traffic requirements, and contention level). To the best of our knowledge, even if the field of Medium Access Control for WSNs is widely studied in the literature, this is the first attempt to address the issue of receiver-oriented dynamic TDMA for WSNs.

The paper has the following organization: Section II overviews the basic concepts of SPARE MAC; in Section III, we describe the proposed solution to dynamically manage the slot assignment procedure, whose performances are thoroughly evaluated in Section IV through simulation. Section V concludes the work.

II. SPARE MAC description

In SPARE MAC, time is organized into frames, and each frame is further divided in a Signalling SUbframe (SSU, consisting of N slots), a Wake-up Slot (WS), and a Data

978-1-4244-4148-8/09/$25.00 ©2009

SUbframe (DSU, made of M slots). The SSU is used for the exchange of all topological and control information needed to manage the network; out of the N slots that make the SSU, each sensor must choose one, which will be used to send broadcast control packets. The Wake-up Slot is used to send out tones to notify neighbouring nodes that they have to wake up during the next SSU, because an exchange of control packets has to take place; in order to ensure that these tones are received, all terminals are always forced to stay active during this slot. Finally, the DSU contains data slots which can be used for the reception of data packets. Each sensor has to choose one or more; those that are chosen form the sensor's Reception Schedule (RS).

So far we haven't considered any rules about which control and data slots can be chosen. A radio network must always deal with the hidden terminal problem; the solution adopted by SPARE MAC is based on the Wake-up Reliable Reservation Aloha protocol [12], according to which each sensor has to choose one of the control slots in the SSU and to use it to transmit signalling packets. The purpose is making these slots collision-free: to achieve this, sensors report which other nodes have transmitted during each control slot. By keeping track of all the data received through these packets, it is possible for a node to find out which slots have been chosen by its neighbours' neighbours; a slot is declared free only if no one is reported using it within 2 hops. This ensures that during the SSU the hidden terminal problem is solved; during the DSU, on the other hand, the rule is different. A slot can be chosen if no other node at 1 hop has chosen it.

There is, therefore, a big difference between control slots and data slots. A sensor uses its control slot to *transmit*; on the contrary, it uses its data slots to *receive*. Each sensor has only one control slot, but its RS can have more than one data slot. A sensor doesn't need to wake up during its control slot, but it has to during its data slots. Finally, control slots are collision-free, since the hidden terminal problem is prevented, while collisions can occur during data slots.

Now we can consider how the network setup is performed. First, a sensor has to acquire synchronization with the other sensors; then it sends a wake-up tone, and keeps doing so until setup is complete. During all this time, it stays awake during the SSU and listens to incoming control packets. Thanks to this, the newcomer can choose a free slot and use it to transmit its own control packet. Then, it waits for a frame: if all neighbours report that during that slot they received its packet, the choice is considered accepted; otherwise, another slot is chosen at random (this can happen if another sensor is joining the network at the same time, and it has chosen the same slot).

Due to collisions and transmission errors, successful reception of data packets can't be guaranteed, thus an explicit Ack is sent in control slots. If the transmitter doesn't receive the Ack, it starts the Collision Resolution Algorithm, which uses binary exponential backoff: the node refrains from transmitting in that slot for a random number of frames between 0 and $2^k - 1$, where k is the number of consecutive collisions experienced by the transmitted packet.

III. Making SPARE MAC Flexible and Dynamic

Leveraging the SPARE MAC concepts described in the previous section, we introduce here novel techniques to dynamically adapt the TDMA structure to the changing nature of the network traffic. Namely, in Section III-A we introduce a dynamic mechanism to reduce the impact of collisions as the traffic grows through a reservation-based approach. Moreover, in Section III-B we describe a dynamic TDMA mechanism that allows sensors to take extra slots when they are needed, and to leave them to reduce consumptions. In fact, in order for the protocol to work, some decisions have to be made about how many data slots to assign to each sensor. Clearly, the more they are, the higher the QoS that can be obtained, but also the consumption. In every situation there is an ideal configuration that makes it possible to meet the QoS requirements, in terms of throughput and delay, while at the same time minimizing consumptions. Our aim is having a protocol that can automatically find the best configuration.

A. RS Reservation

In order to avoid the long delays caused by collisions and the subsequent exponential back-off, we propose here a dynamic technique to reserve reception slots in SPARE MAC. This prevents collisions from happening, thus making it a contention-free protocol.

The basic idea is that if, for example, a sensor has 2 slots, and exactly 2 sensors are contending them, we want to assign one slot to each of them, so that collisions are prevented. Unfortunately, the problem of finding a way to share any number of slots among any number of transmitters is often non-trivial: in the example we've just seen, if the sources were 3, one of them wouldn't be able to have a reserved slot. The solution consists in introducing a multiframe structure, whose period is equal to the number of sources. In this 3-frames multiframe there are overall 6 slots, and it is easy to assign 2 of them to each of the 3 sources, each one ending up with a capacity equivalent to 2/3 of a slot. By doing this it is always possible to reserve a minimum bandwidth to each source. Of course, the rate of some sources may be lower: in that case, more slots can be assigned to other sources, with a higher data rate. The concept is that every source has the right to have a fair share of the available bandwidth, and only if it doesn't need it a part can be assigned to other nodes. Slot periods don't need to be the same, so that grouping is possible: for example, slot 1 can be assigned only to sensor A (with period 1), while slot 2 can be shared by A, B and C (with period 3).

The procedure is triggered when a sensor detects too many (default value is 3) consecutive collisions towards the same node; the request is issued by transmitters, but the assignment is decided entirely by the receiver. The task of transmitters is therefore simple: they only have to count consecutive collisions, and if the threshold is reached they send a control packet containing a request. On the contrary, the task of the receiver is more complex. First, a rate estimation algorithm is needed: it allows the receiver to know how many sources are active (by measuring how much time has passed since

978-1-4244-4148-8/09/$25.00 ©2009

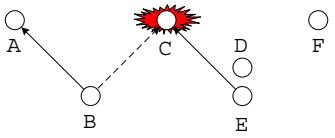

Fig. 1. An example of problem given by hidden terminals.

the reception of the last packet) and how much traffic they are sending. Then, the number of active sources becomes the multiframe period, while their rates, together with the requests, are used to choose how many slots must be reserved to each transmitter. When this is done, a control packet is sent; a special field makes it possible to specify the slot we are considering, its multiframe period, and, for each of the frames, to which sensor the slot is exclusively reserved.

The rate estimation algorithm works by having each sensor store the inter-arrival times of the last 50 packets it receives, calculating the average and standard deviation. This makes it possible to evaluate the average rate of incoming packets, measured in [packets/frame]. Chebyshev's inequality[1] is used to determine when an active source becomes inactive: if the time that has passed since the last packet was received exceeds a threshold, the source is declared inactive. This threshold is the sum of the average and seven times the standard deviation; according to Chebyshev's inequality, the probability that an inter-arrival time is higher than this is at most 1/49. Moreover, if incoming traffic is Poisson-distributed, inter-arrival times follow a negative exponential distribution, and the corresponding cumulative distribution function is $F_X(x) = 1 - e^{-\lambda x}$. From this, and remembering that both the average μ and the standard deviation σ are $1/\lambda$, we can easily calculate that the probability that an inter-arrival time is above the threshold of $\mu + 7\sigma$ is $e^{-8} \approx 0.0003$, which is a negligible value. It is therefore very unlikely to consider a source inactive when actually it is still active.

A problem occurs with some topologies. For example, let's observe Figure 1. If no reservations are used, some collisions may occur due to the hidden terminal problem: in this example, sensor B is sending a packet to A, but its transmission is also heard by C, which is receiving another packet from E, and therefore C experiences a collision. We might think that reservations solve the problem, but unfortunately this can lead to a deadlock if two conflicting reservations are sent, because a sensor could receive the authorization to transmit from one of its fathers, but at the same time the authorization could be denied by another father[2]. In Figure 1, if C authorized E but during the same slot F authorized D, neither D nor E would be able to transmit. Having nodes coordinate and synchronize assignments looks like a solution, but it would be long and difficult to manage, forcing nodes at 2 hops from each other to continuously reschedule assignments, therefore it is better to solve the problem by taking a new approach, that is, changing the slot reuse rule: the new one states that if a node's father chooses a certain slot, all the other neighbours of the node must avoid that slot, and vice versa. The result is that whenever a packet is transmitted, there is always only one receiver, the intended one, and other nodes can't be involved in a collision. Unfortunately, this rule requires some extra signalling, because it forces sensors to coordinate themselves with neighbours at 2 hops when choosing their RS. Every sensor reports which slots have been chosen by its neighbours, but this isn't enough: if 2 sensors decided to choose the same slot at the same time, they wouldn't have a way to find it out, as they would both expect to see their neighbours to report it. This means that an additional field is required, used to indicate 2-hop conflicts. But again, this is not enough: so far, 2 nodes would be able to realize they have been involved in such a conflict, but they need some coordination to solve it. The winner is chosen by common neighbours: fathers have priority over brothers, which in turn have priority over sons; in case the conflicting nodes have the same depth, the one with the lowest ID is chosen. The winner keeps its slot, while all the others have to leave it. This change to the slot reuse rule implies that the network setup must also be modified: a father can't start choosing its RS until it receives an explicit authorization by all of its sons. In Figure 1, B authorizes A, while D and E authorize C (the same rule as before is applied, that is, the father with the lowest ID is chosen); A can choose its slot, but C can't, because one of its sons, that is B, forbids it. When A finishes it will broadcast its RS; B will authorize C and at the same time mark the slot chosen by A as unavailable, and C will choose a different one. After that, C will broadcast its choice, and both D and E will authorize F, while informing it on the slot chosen by C; F will choose a different one. It is possible for F to choose the same slot taken by A, as this doesn't cause any problems.

This new setup procedure is therefore slower, but on the other hand it takes place only once, and it makes it possible to completely avoid the hidden terminal problem, and thus reservations can be used successfully.

B. Dynamic Bandwidth Adjustment

The purpose of this feature is to make it possible for any sensor to dynamically change its own RS, by adding, removing, or changing some of its slots. Moreover, it is sometimes necessary to issue requests to some neighbours. This requires control packets to be sent, and even though control slots are collision-free, the poor quality of radio channels implies that an explicit acknowledgement must be sent every time. Each of these actions requires a specific field in control packets. As usual, a wake-up tone must be sent before doing this, in order to ensure that the packet will be received. The protocol must therefore allow each sensor to:

- Make a request
- Inform neighbours on its own new RS
- Acknowledge changes to a neighbour's RS

The first and third field only require one bit for each possible neighbour: all we need is being able to say if we are (bit 1) or are not (bit 0) making a request, and if we are (1) or are not (0) confirming the reception of a signalling message. Both

[1] $P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$, regardless of the distribution of the random variable X, where μ and σ are its average and standard deviation.

[2] By "father" we simply mean a next-hop node, that is, a neighbour that is closer to the sink; in this sense, a sensor can have more than one father.

978-1-4244-4148-8/09/$25.00 ©2009

fields always include one bit for each neighbour, thus making it possible to take part into as many signalling procedures at the same time as is needed. The second field requires one bit for every data slot; bit 1 means that the corresponding slot was chosen and the sensor will wake up during that slot for reception, while bit 0 means that it will stay in sleep mode; any number of slots can be added, moved, or removed. As we've seen, the protocol allows to apply any change to the RS, managing it in complete freedom. Now we need to develop an algorithm that decides when to perform these actions.

1) RS Increase: Increasing a sensor's RS has 2 purposes: it can be done to increase the bandwidth and keep up with the rate of data sources, or to lower the delay experienced by received packets. The delay reduction is twofold: if the transmitter has other packets waiting to be sent, having more slots means the queue is shorter; on the other hand, if there are no other packets to be sent, having more slots simply means that they are more frequent and therefore the wait for one in which to forward a packet is shorter.

Now let's consider under which circumstances the Increase procedure is called. A sensor can respond to a request issued by another node, or choose to start it on its own. In the first case, each sensor tries to evaluate the delay that will be experienced by the packets it will send, and this estimate is based on 2 measures: the average service time and the average queue length. The service time is defined as the duration of the interval between the first transmission of a packet and the reception of the corresponding Ack, rounded up to reach an integer frame (the idea is that a slot can, at its best, be used to transmit no more than a packet per frame, so it isn't acceptable to have a service time that is less than a frame). The queue length is measured at the beginning of every frame. Both these estimates are averaged using the formula

$$AVG_{curr} = \alpha \cdot VAL + (1 - \alpha) \cdot AVG_{prev}$$

where the value AVG_{curr} is the current average, AVG_{prev} is the previous average, and VAL is the current value; we use $\alpha = 1/8$. The average delay that we expect a packet will experience is given by the product of the average service time by the average queue length; if it goes beyond a specified Target Delay, a request for an extra slot is made.

The other situation in which the Increase procedure is called is the case in which a sensor autonomously decides to increase its own RS because the average delay it measures is too high. Every time a node receives a packet, it checks the timestamp indicating when it was received by the previous node, and calculates the delay experienced for that hop. If on average this value is higher than the Target Delay, the Increase procedure is called.

One final thing that is noteworthy about the Increase procedure is that it is forbidden for a sensor to take more than half of the total data slots, as this would leave its neighbours with very few slots.

2) RS Decrease: The Decrease procedure is the opposite of the Increase: its purpose is to choose one or more slots to drop and to inform all neighbours. It is very similar to the Increase procedure; only, it is always started autonomously by the receiver.

Deciding when to call the Decrease function is a difficult task. In fact, we must keep in mind that we are generally trying to reach a certain level of QoS, in terms of throughput and delay, while minimizing power consumption. In case of the Increase procedure, it is easy to notice that the QoS that is being provided is not enough, and therefore choosing to increase the RS is necessary: it will cause power consumption to increase, but it is inevitable. On the other hand, assuming that the QoS level is satisfying, how can we determine whether by dropping a slot (which allows for energy saving) the QoS requirements will still be met? As far as throughput is concerned, this is quite easy to establish, as we'll see, but dealing with the delay constraint is a lot harder. Let's focus on the first problem, that is, detecting when the current RS is excessive for the amount of traffic that the node is receiving. A counter is associated with each slot, and it keeps track of how many frames have passed since the last time the node has received a packet during that slot. The counter is reset when a packet is received. Under these conditions, if a certain Inactivity Threshold is reached (the chosen value is 70 frames), the Decrease procedure is invoked. Whenever the Decrease procedure is called, for any reason, all counters are reset, so that the counters always refer to the inactivity of the slots when they are that many. The other case in which the Decrease procedure is started is the one in which the measured delay is too low. The difficult task is understanding when the delay is so low that by dropping one (or more) slots the Target Delay constraint won't be violated. Unfortunately, coming up with an exact formula capable of predicting the consequences of such an action is extremely difficult, as the system is neither linear nor deterministic, due to backoff, and it is also time varying. We use hereafter the following practical approximation: by passing from X to $X-1$ slots, the delay is expected to increase by a factor of $X/(X-1)$.

IV. Performance Evaluation

SPARE MAC has been implemented in Network Simulator 2 to test its performance. To ensure that all our results are comparable, we've used the same test scenario in all the simulations. It consists of a binary tree with 15 sensors: the sink at the top (depth 0), 2 sensors at depth 1, 4 sensors at depth 2, and 8 leaves at depth 3, which are the sources of data transmissions. The topology is shown in Figure 2; the following parameters have been used: N = 15, M = 20, control/wake-up/data slot length = 100/9/560 bytes, Poisson traffic, 8 sources, power consumption in TX/RX/sleep = 24.75 mW/13.5 mW/5 μw, duration = 1000 seconds, bit rate = 250 kbit/s, target delay = 0.27 seconds (per hop); we vary the amount of offered traffic per source from 125 bit/s to 2000, by steps of 125; since there are 8 sources, the traffic across the whole network ranges from 1 to 16 kbit/s. All tests have been repeated 50 times and each measure has been averaged. The network is small, but on the other hand the number of nodes is irrelevant: the protocol is entirely distributed, therefore it

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

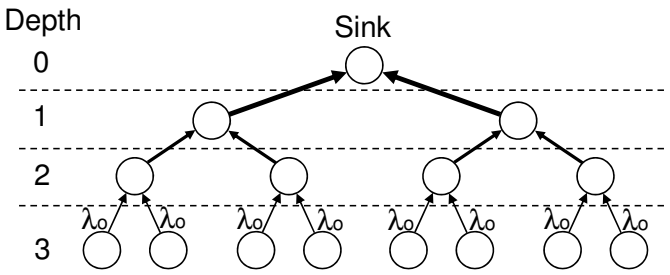

Fig. 2. The network topology used in simulations.

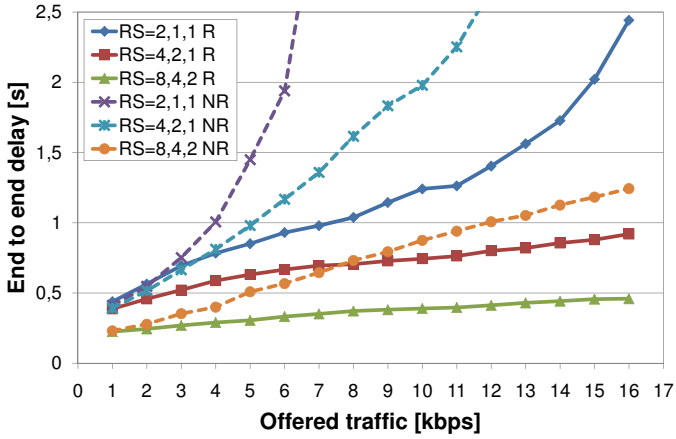

Fig. 3. End-to-end delay versus offered traffic, with and without reservations. In the legend, R stands for Reservation and NR for No Reservation.

scales well, since the only thing that matters is the number of neighbours at 1 and 2 hops.

To evaluate the performance of the protocol, we consider 2 results: the average end-to-end delay and power consumption. Since the number of slots plays a fundamental role with regard to both, we must consider different configurations separately. We have selected 3 different RS assignments; "RS = 4,2,1" means that the node at depth 0 (the sink) has 4 slots, those at depth 1 have 2 slots, and those at depth 2 have only one.[3] In Figure 3 we compare the end to end delay obtained with reservations and without them, and in Figure 4 we do the same for power consumption. As we can see, everything works as expected: without reservations the delay increases with traffic and decreases with the number of slots, while power consumption shows a strong linear dependence with traffic, and increases with the number of slots. It is noteworthy that beyond 10 kbit/s the consumption for the 2,1,1 RS increases less than below that threshold. This happens because there are so many collisions (and thus backoffs) that transmissions actually become rare: clearly, this configuration is inadequate for these rates. The use of reservations, on the other hand, can improve both results in a way that increases with traffic, which is reasonable, since higher traffic values cause more collisions (and retransmissions), and the reservation algorithm prevents them, thus its effectiveness increases under heavy traffic. We can conclude, in other words, that the reservation mechanism improves both delays and consumptions.

Now we can consider the dynamic RS. Once again, we

[3]Sensors at depth 3, the leaves, do not receive traffic, thus the number of slots assigned to them is irrelevant; by default they have 1.

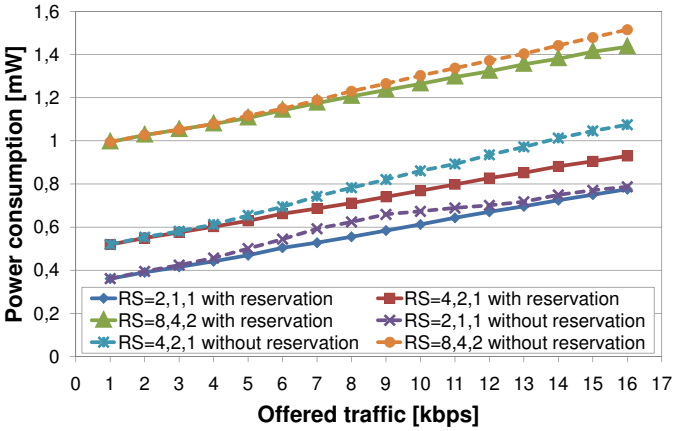

Fig. 4. Power consumption versus offered traffic, with and without reservations.

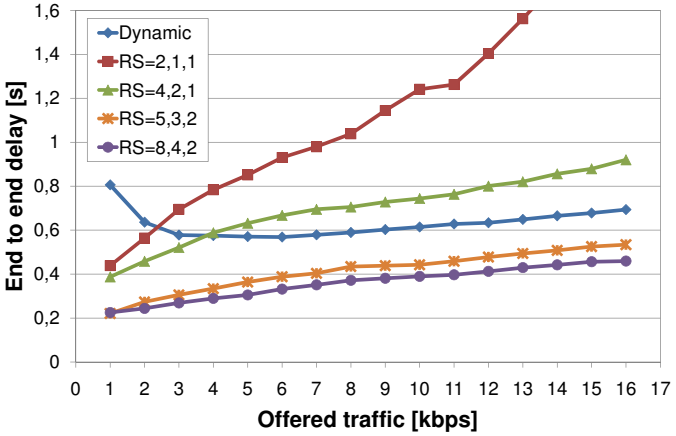

Fig. 5. End-to-end delay versus offered traffic for different static and dynamic RS configurations, using reservations.

want to measure delay and consumption. In Figures 5 and 6 we see a comparison of delay and consumption between different configurations: some cases of static RS (we've also added RS = 5,3,2) and the dynamic one. The 2,1,1 and 4,2,1 configurations fail to keep below the target delay of 0.81 seconds as traffic increases, thus they can't be used in a generic situation where the amount of traffic that will be generated is unknown. On the other hand, using more slots makes it possible to satisfy the delay constraint, but consumption is higher. The dynamic algorithm can always reach its goal, and the consumption is lower than static RSs with many slots. Therefore, it is suitable for situations in which the traffic rate is unknown. We must notice, however, that it requires more energy than static configurations; there are 2 reasons behind this. One is that the dynamic algorithm needs signalling traffic, which requires energy; the other is that, due to the distributed nature of the adaptive algorithm, it is impossible to reach an optimal configuration. The protocol doesn't consider end-to-end delay, but only the hop-by-hop one: each sensor can only try to reduce the delay associated with its hop, it can't interact with the rest of the route. If we want to reach an end-to-end target delay of 3 seconds in a 3-hop scenario, we must configure each sensor to keep the delay below 1 second. If the delays were actually 2, 0.5 and 0.5, the application wouldn't notice it, since the sum would be within the acceptable range,

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

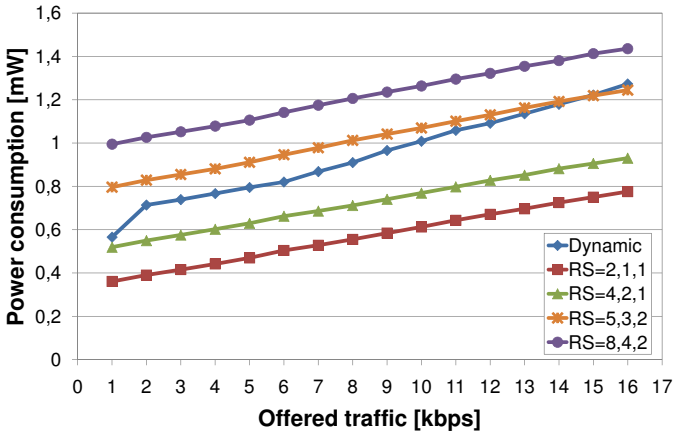

Fig. 6. Power consumption versus offered traffic for different static and dynamic RS configurations, using reservations.

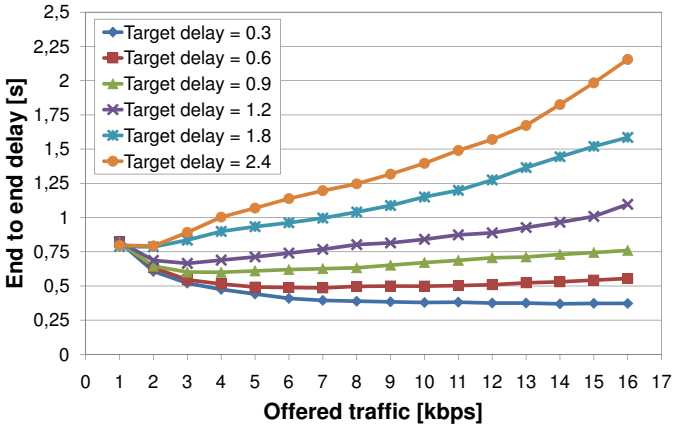

Fig. 7. End to end delay versus offered traffic for different target delay values, using reservations.

but the sensors at depth 2 wouldn't be aware of this, and so they would try to reduce the delay, by taking more slots. This explains why the consumption is sometimes unnecessarily high. Moreover, there's an automatic control problem. The faster and more sensitive is the control signal (i.e.: the smaller the number of packets used to calculate the average delay), the higher is the probability to incorrectly estimate the number of required slots. On the other hand, increasing the number of packets used to calculate the average gives a longer reaction time, and in the meanwhile it is possible that other sensors choose to change their RS, therefore influencing the measures done by the other sensors; a tradeoff between stability and fast reactions must be found.

Finally, we want to show the sensitivity of the algorithm to different target delay values. In Figure 7 we see the average end-to-end delay obtained by specifying different target values. Apart from the most stringent value, that is 0.3 seconds, the algorithm can always keep below the desired threshold. Figure 8 shows the power consumption; clearly, the lower the specified target, the higher the number of required slots and thus the consumption.

V. Conclusions

In this paper, we have proposed dynamic algorithms to manage the resource (slots) assignment in a receiver oriented

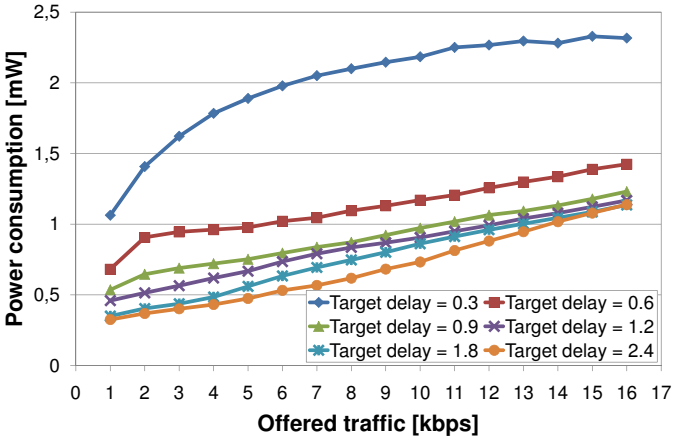

Fig. 8. Power consumption versus offered traffic for different target delay values, using reservations.

TDMA protocol for WSNs, the SPARE MAC. The new features include support for dynamic reservation of data slots to specific transmitters, in order to avoid collisions, and dynamic assignment of reception schedules. A thorough performance evaluation through simulation was carried out to assess the quality of the proposed dynamic algorithms. Results show that reservations can greatly reduce end-to-end delays, and also slightly improve energy consumptions, whereas the dynamic TDMA algorithms makes it possible to adapt to different traffic conditions, while guaranteeing loose QoS constraints.

References

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

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

[3] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 18–39, Fourth Quarter 2008.

[4] A. B. Mahmood Ali and M. Jonsson, "Wireless sensor networks for surveillance applications - A comparative survey of mac protocols," *The Fourth International Conference on Wireless and Mobile Communications*, pp. 399–403, July 2008.

[5] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated, adaptive sleeping for wireless sensor networks," *IEEE Transaction on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[6] V. Cionca, T. Newe, and V. Dădârlat, "TDMA protocol requirements for wireless sensor networks," *SENSORCOMM '08*, pp. 30–35, 2008.

[7] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wireless Networks*, no. 12, pp. 63–78, 2006.

[8] C. D. Young, "USAP: a unifying dynamic distributed multichannel TDMA slot assignment protocol," *in Proc. IEEE MILCOM '96*, vol. 1, October 1996.

[9] V. Turau and C. Weyer, "Scheduling transmission of bulk data in sensor networks using a dynamic TDMA protocol," *International Conference on Mobile Data Management*, pp. 321–325, May 2007.

[10] A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio, "Dynamic TDMA slot assignment in ad hoc networks," *Proceedings of the 17th International Conference on Advanced Information Networking and Applications (AINA03)*, pp. 330–335, 27-29 March 2003.

[11] L. Campelli, A. Capone, M. Cesana, and E. Ekici, "A receiver oriented MAC protocol for wireless sensor networks," in *proc. of IEEE MASS 2007*, Pisa, Italy, October 8-11 2007, pp. 1–10.

[12] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, "ADHOC MAC: new MAC architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services," *ACM Wireless Networks*, vol. 10, no. 4, pp. 359–366, July 2004.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.