A Distributed Feedback Control Mechanism for Quality-of-Service Maintenance in Wireless Sensor Networks

(Extended Abstract)

Marcel Steine*, Marc Geilen* and Twan Basten*†
*Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands
†Embedded Systems Institute, Eindhoven, The Netherlands
Email: {m.steine,m.c.w.geilen,a.a.basten}@tue.nl

***Abstract*—Wireless sensor networks are typically operating in a dynamic context where events, such as moving sensor nodes and changing external interference, constantly impact the quality-of-service of the network. We present a distributed feedback control mechanism that actively balances multiple conflicting network-wide quality metrics, such as power consumption and end-to-end packet latency, for a heterogeneous wireless sensor network operating in a dynamic context. Nodes constantly decide if and how to adapt controllable parameters of the entire protocol stack, using sufficient information of the current network state. Using experiments with an actual deployment we show that our controller allows to maintain the required network-wide quality-of-service, with up to 30% less power consumed, compared to the most applicable (re-)configuration approaches.**

I. Motivation

Controllable parameters of individual sensor nodes in a Wireless Sensor Network (WSN), such as the radio transmission power, MAC protocol duty cycle and selected routing parent(s), determine the behaviour of the network. The values of these parameters, referred to as the *configuration*, determine to what extent expectations on quality metrics, i.e., the required Quality-of-Service (QoS), are satisfied. WSNs typically operate in a dynamic environment or exhibit dynamic behaviour themselves causing the configuration required to achieve sufficient QoS to vary over time. Statically configured nodes [5], [6] cannot cope with dynamics; run-time adaptation of parameters, or reconfiguration, is needed. Current approaches focus either on a single metric [4], [16], typically energy consumption, or on optimizing local metrics [13], [18] and often consider a homogenous network. In practice, the requirements of the applications are typically expressed by multiple conflicting quality metrics expressed on a network-wide scale [3], [14], such as a maximum end-to-end latency of 100 milliseconds, a packet delivery ratio of 90% or maximal network lifetime. There is only limited existing work on run-time adaptation techniques that consider multiple QoS metrics expressed on a network-wide scale. The recent work of [19] proposes a centralized approach which determines how to adapt MAC protocol parameters based on centrally collected network QoS information. In practice, centralized approaches might be too costly, non-scalable or not able to respond quick enough to dynamic events. By focusing on only the MAC protocol, it does not exploit the fact that the parameters from all protocols may influence the behaviour of the network [7]. As a homogenous configuration is assumed where all nodes use the same MAC parameters, it furthermore ignores the typical heterogeneity in WSNs and resulting variation of the impact of nodes on the trade-offs involved.

In this extended abstract, we introduce a distributed feedback control mechanism to maintain a required QoS, defined by multiple quality metrics, for a WSN in a dynamic context. The approach separates the adaptivity from protocol operation and lets nodes determine if and how they should adapt controllable parameters of the entire protocol stack. They do so based on local estimates of the current (global) network QoS and expected impact of parameter adaptation. This information, i.e., the feedback, is efficiently propagated through the network using the distributed service of [12]. We show that we can successfully use our mechanism to maintain a required level of QoS for an actual dynamic and heterogeneous deployment.

Section II describes our feedback control mechanism. In Section III we explore the behaviour and performance of the controller in more detail and Section IV concludes.

II. Distributed Feedback Control Mechanism

We start this section with a motivating example to intuitively show the need and required functionality of our controller. The rest of the section describes the feedback control mechanism in more detail.

A. Motivating Example

Fig. 1 shows a simplified example of a WSN with 9 nodes. The edges represent the communication links to forward packets to sink S. Next to the links we show the latency of the link together with the power spent in the current configuration of the sending node. Every node takes part in one or more paths from some node to the sink. Adaptation of parameter values of a node will therefore not only influence its local power consumption and link latency to the parent, but also the end-to-end latency of all of these paths. After dynamic changes, three situations may arise for which the current configuration cannot meet an end-to-end latency of 1000 milliseconds, while minimizing the maximum power consumption.

(1) The end-to-end latency of a path is (much) higher than the target, e.g., from node G to the sink. Additional power should be spent to reduce the latency of the path. Multiple nodes, C, D and G, can adapt to achieve this. Since we want

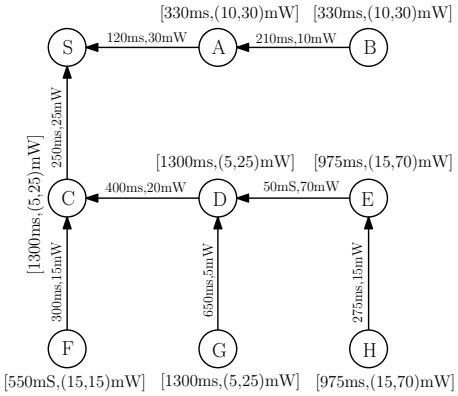


Fig. 1: Different situations violating QoS requirements

to minimize the maximum power consumed over all nodes, it is preferred that nodes with a lower power consumption adapt earlier than the nodes with a higher power consumption. G is preferred to adapt first. With local information on the minimum and maximum power on the (maximum-latency) path, nodes can easily decide whether they are consuming a higher power compared to others on the path.

(2) The end-to-end latency of a path is (much) lower than the target. This could signal an excessive use of power. Parameter adaptation where power is reduced at the expense of a higher latency might be possible. For the low latency path from node B to the sink, nodes A and B can adapt to exploit this trade-off. Now it is preferred to first adapt node A with the highest power consumption. The latency from node F to the sink is also much lower than the target, but node C on that path is unable to increase its link latency (in exchange for power) as it is also part of a high-latency path. This should be reflected in the minimum and maximum power information of node F, such that it locally knows that, in this case, it is the node with the highest power consumption able to adapt the path; the minimum and maximum power are determined based on collected power consumption of the nodes which have the same maximum-latency path.

(3) The latency is close to the target, but the power consumption on the path is not minimized. Due to the discrete nature of the parameter adaptation it might be possible that the end-to-end latency will never be exactly the target latency; therefore we have to define what is close enough. For example, a value within 10% of the target latency. On the path from H to the sink, node E uses significantly more power than H. A better balancing of power would allow us to reduce the power used by node E and thereby the maximum on that path. This could be achieved by high power node E reducing its power (increasing the latency of the path), while low power node H increases its power (reducing latency).

Local knowledge at a node i of the maximum end-to-end latency for all paths through node i, $e2elat_i$, and minimum and maximum power on that maximum-latency path, $minP_i$ and $maxP_i$, are assumed to be available by using an existing generic distributed service to efficiently maintain local estimates of minimum-cost path information for all nodes in a dynamic WSN [12]. Using repeated broadcasts, local knowledge is iteratively derived based on information from neighbouring nodes. We assume that the service is configured such that the feedback is propagated as fast as possible for the given deployment scenario, keeping in mind the trade-off with the overhead in terms of extra communication of packets. More details can be found in [12].

In the remainder of this section, we introduce our distributed feedback control mechanism. For ease of explanation we focus on typical QoS goals, namely meeting a constraint on the end-to-end latency of packets from a node to a sink (or cluster-head or any other point of interest), while minimizing the maximum (average) power used by any of the nodes (which correlates well with network lifetime). The control mechanism is generally applicable to other trade-offs as well. For example, the controller can focus on end-to-end packet delivery ratio, or a more elaborate definition of lifetime can be used which uses estimates of residual power.

B. Distributed Feedback Control

Our distributed feedback control mechanism consists of two steps. First, a desired 'rate of change' at which the link latency of the node should be adapted is computed based on the maximum end-to-end latency of the paths through the node. The rate defines the (positive or negative) change in latency (in seconds) a node is expected to achieve every time unit, e.g., second, by adapting its parameters. The rate is negative if the node needs to reduce the latency of the path and positive if the node can increase the latency to approach the target latency. This rate may differ per node according to the considerations as discussed in the beginning of this section. Second, after the rate is determined, the node translates it to new values of the parameters that it should adapt.

a) Determining rate: The rate of change, $rate_i$, for node i when the end-to-end latency is not close to the target, i.e., situations (1) and (2), is determined as follows.

$$offset_i = targetLatency - e2elat_i$$

$$b = \begin{cases} \dfrac{pow_i - minP_i}{maxP_i - minP_i} & \text{if } minP_i \neq maxP_i \\ 0.5 & \text{if } minP_i = maxP_i \end{cases}$$

$$rate_i = \begin{cases} \dfrac{offset_i * (1-b)}{k} & \text{if } offset_i < 0 \\ \dfrac{offset_i * b}{k} & \text{if } offset_i > 0 \end{cases}$$

First, we set the rate proportional to the difference between the target latency, $targetLatency$, and the maximum end-to-end latency of node i, $e2elat_i$. Furthermore, we scale the rate based on the local (average) power currently spent, pow_i, and the minimum, $minP_i$, and maximum, $maxP_i$, power of the nodes adapting for the same maximum-latency path. In case the path latency needs to be reduced, i.e., $offset_i < 0$, the nodes with a lower power have a higher rate and therefore larger impact on their power than nodes consuming more power. The same, but reversed, holds if the path latency can be increased. In case a node locally believes that all nodes on the maximum-latency path are using the same power level, i.e., $maxP_i = minP_i$, the same average rate is expected to be achieved from all nodes. With an additional parameter $k > 0$ we adjust the overall speed, or loop-gain, of the controller.

The speed of the controller has an important influence on the behaviour of the controller, such as stability. An unstable solution, where the latency drifts away from the target latency, should be avoided. The speed of the controller furthermore plays a role in the trade-off between the accuracy of the controller in providing the required latency and the speed of converging to the target. Simulations, using discrete-event simulator OMNeT++ [10] with WSN modeling framework MiXiM [8] show that with proper selection of the value of k the controller can successfully settle the value of the latency within the given bounds after the impact of a single dynamic upset event, known as the step-response. In practice we continuously have big and small changes in the quality metrics of the WSN due to dynamic events and there is no single static optimal configuration over time. The behaviour of the controller in a practical set-up is studied in Section III.

As soon as the maximum observed end-to-end latency is within the target range, e.g., situation (3), the rate is calculated with the goal of balancing the power. This is done as follows.

$$rate_i = \frac{m * dist_i}{k} \ , \ dist_i = 2\frac{pow_i - minP_i}{maxP_i - minP_i} - 1$$

The value of $dist_i$ for node i is -1 if the node's power consumption is equal to the minimum used on the path and 1 if it is the maximum. To determine the rate of adaptation/balancing for a given node, we multiply $dist_i$ with a predefined (maximum) rate at which we want to balance, m. This parameter allows us to influence the granularity of the reconfiguration used to balance the power consumption. In practice we do not want the maximum balancing step to be too large, say maximal 10% of the target QoS metric, to avoid large overshooting of the controlled metric. The rate is furthermore scaled by k to influence the adaptation speed.

b) Reconfigure based on calculated rate: To achieve the calculated rate of change, nodes periodically translate the rate into a new configuration by adapting parameter values. The length of such a $round$ determines the maximum frequency of adaptation. As the adaptation of parameters can typically be done instantaneously and without significant overhead, rounds are preferably short. Every round, a node decides whether an adaptation is needed, and if so, which new configuration to use. For this, nodes estimate the impact on the local link latency, and thereby end-to-end latency, that a particular adaptation has. In the current implementation, we use simple linear estimation models. The rate can then be translated to the required reconfiguration. Because of the discrete nature of the parameters the controller uses a probabilistic approach to decide whether to reconfigure in a given round. For example, with a required adaptation of 10 milliseconds and a minimum impact of reconfiguration of 100 milliseconds, the probability of adaptation is set to 10%. The same approach is used when the rate is between two possible reconfiguration options.

Feedback control does not depend on accurate estimates of the expected impact as long as it is known that a particular adaptation will increase or decrease a particular quality metric. If the impact is not as expected this will be observed in the feedback and the controller will continue to steer the latency to the target. Integrating more knowledge to model the expected impact, for example using online learning, could potentially increase the accuracy of the estimation and the overall accuracy of the controller and is part of future work.

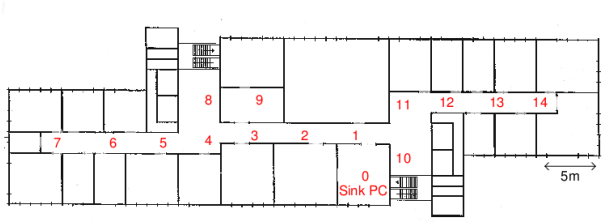
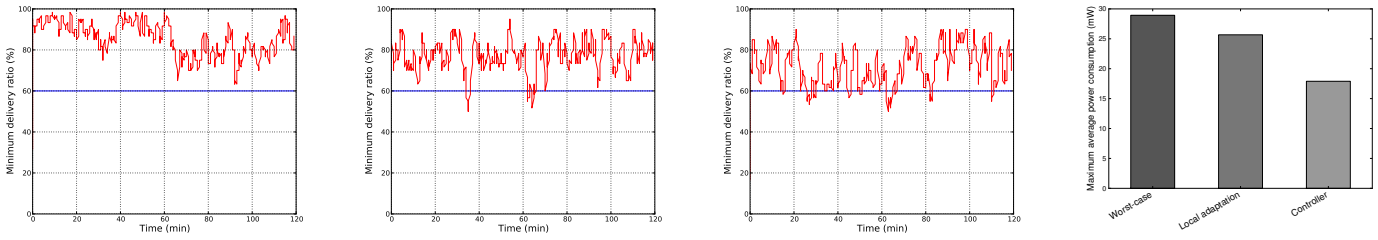


Fig. 2: Deployment set-up static nodes

III. Controller Performance Analysis

Using the controller, all nodes repeatedly calculate a rate and adapt their parameters. During this distributed control process many actions happen at the same time, including the probabilistic discrete adaptation of parameters and unreliable, stochastic feedback propagation over the wireless channel. To analyse the performance of our controller in practice, we implemented a controller, together with the distributed service to propagate feedback information, in TinyOS [17] to maintain a given delivery ratio while minimizing the power for a WSN deployed in our office building. It consists of 15 static TelosB [15] nodes deployed as shown in Fig. 2. In addition, 5 persons, which occasionally move between different locations in the office building, have a BSN [2] sensor node attached which sends application packets at a rate of one packet every second to the sink (node 0), using the static nodes when needed. The MAC protocol we consider for all nodes is the default Low-Power-Listening MAC [9] available in TinyOS. We use an approach inspired by [1] where we always retransmit a packet a given number of times and not rely on the receiver to send acknowledgments. Static nodes use a fixed static node to route data to. Mobile nodes send packets to the neighbouring node which gives the highest delivery ratio to the sink (which is locally known with the use of the service propagating the feedback). The protocol stack has several controllable parameters and we focus on two of them. First, we have the transmission power of the radio [15]. Second, we look at the number of retransmissions of the MAC protocol, being either 0, 1 or 2. Based on experiments with the deployment, we estimate a single step in transmission power to improve/reduce the end-to-end delivery ratio by 10%. For the retransmission number, we consider the value to change the delivery ratio by 5%. We give priority to adapting the transmission power, as this is expected to give the largest impact in power consumption.

We configured the controller such that it aims at balancing the delivery ratio between 60% and 70%. Based on small-scale experiments, we set the length of a round to 10 seconds, k to 30 and m to 5%. We compared the performance of our controller with a typical approach where a single configuration anticipating for the worst-case is used. Additionally, we compare with what we believe is the best existing run-time adaptation approach. As there exists no distributed approach which considers adapting multiple parameters at the same time, while controlling multiple network quality metrics, we construct our own. It uses the same information on the impact of parameters but, instead of adapting based on network QoS,



(a) (b) (c) (d)

Fig. 3: Minimum delivery ratio for (a) worst-case, (b) local and (c) controller approach. (d) Power consumption per approach

i.e., the end-to-end delivery ratio, it adapts in order to steer local QoS, i.e., the delivery ratio to the immediate parent. The number of hops to the sink determines the minimum delivery ratio every link should have to meet the constraint of 60%. As a local approach does not consider this global information, we have to make assumptions on this. We assume a maximum number of hops of 4 to reach the sink and therefore set the target delivery ratio of every link to 90% in order to meet the overall delivery ratio target of at least 60%.

Fig. 3 shows the minimum delivery ratio over all nodes in the network for a 2 hour experiment for the three approaches. The values are determined every 10 seconds based on the number of packets received at a TelosB sink connected to a PC in the last minute, knowing that the mobile nodes send one packet every second. Fig. 3d shows the resulting maximum average power of any of the nodes in the network. The average power spent by a single node during the experiment is based on the power consumed by the radio, the most dominant part of the power consumption [11]. It is determined by locally monitoring the time spent in every radio state, i.e., sending, receiving and idle [15], and the transmission power that is used. Note that this measurement of power consumption takes all the packet communication into account, including the additional overhead incurred by the service used for the propagation of QoS information.

Based on these experimental results we can make several observations. First of all, we have shown that our controller is simple enough to be implemented on the resource constrained nodes and can successfully respond to changes in network QoS in a distributed manner. Compared to a single worst-case configuration approach, we see a clear improvement in power consumption, taking into account the overhead of feedback propagation for our controller; as we avoid QoS over-provisioning for the non worst-case situations. The worst-case approach has fewer violations of the delivery ratio constraint and fewer fluctuations in delivery ratio, mainly due to the large over-provisioning. In practice we do tolerate occasional violations of the constraint in exchange for the significant reduction in power consumption.

The local approach has a delivery ratio behaviour similar to our controller, but an about 30% higher maximum average power is required. It does not consider knowledge of the current state of the system, such as the actual number of hops a packet has to travel, and the heterogeneity in the QoS provisioning. This results in an unbalanced power consumption, resulting in a maximum average power consumption which is higher than for our distributed controller.

IV. Conclusions

In this extended abstract we introduced a distributed feedback control mechanism that actively maintains the required QoS of a WSN. Nodes adapt their parameters directly based on the estimates of the current network QoS. Experiments with an actual deployment show that we are able to implement the controller on resource constrained nodes and to maintain an appropriate level of QoS for a dynamic and heterogeneous deployment. Compared to a worst-case single configuration and a local adaptation approach that does not take network QoS knowledge into account, we are able to maintain the required level of delivery ratio at a lower maximum average power consumed by nodes in the network.

References

[1] M. Blagojevic et al. Probabilistic Acknowledgment Mechanism for Wireless Sensor Networks. In *NAS*, pages 63–72, 2011.

[2] BSN website, ICL London. http://vip.doc.ic.ac.uk/bsn/m621.html.

[3] M. Ceriotti et al. Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment. In *IPSN*, pages 277–288, 2009.

[4] O. Chipara et al. Real-time Power-Aware Routing in Sensor Networks. *IWQoS*, pages 83–92, 2006.

[5] K. P. Ferentinos and T. A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4):1031 – 1051, 2007.

[6] R. Hoes et al. Quality-of-Service Trade-off Analysis for Wireless Sensor Networks. *Performance Evaluation*, pages 191–208, 2009.

[7] T. Melodia, M. Vuran, and D. Pompili. The State of the Art in Cross-Layer Design for Wireless Sensor Networks. In *Wireless Systems and Network Architectures in Next Generation Internet*, pages 78–92, 2006.

[8] MiXiM website. mixim.sourceforge.net.

[9] D. Moss and P. Levis. BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking. Technical report, Stanford Information Networks Group Technical Report, 2008.

[10] OMNeT++ website. www.omnetpp.org.

[11] A. Prayati et al. A modeling approach on the TelosB WSN platform power consumption. *J. Syst. Softw.*, 83:1355–1363, August 2010.

[12] M. Steine, M. Geilen, and T. Basten. Distributed Maintenance of Minimum-cost Path Information in Wireless Sensor Networks. In *PM2HW2N*, pages 25–32, 2011.

[13] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *SenSys*, pages 1–14, 2008.

[14] R. Szewczyk et al. An analysis of a large scale habitat monitoring application. In *SenSys*, pages 214–226, 2004.

[15] TelosB Datasheet, Crossbow Inc. www.xbow.com.

[16] H. Tian et al. SPEED: a stateless protocol for real-time communication in sensor networks. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 46 – 55, 2003.

[17] TinyOS website. www.tinyos.net.

[18] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 13(3):493–506, 2004.

[19] M. Zimmerling et al. pTunes: Runtime Parameter Adaptation for Low-Power MAC Protocols. In *IPSN*, 2012.