Middleware Support for Adaptive Real-Time Applications in Wireless Sensor Networks*

João Alves, António Casimiro, and Luís Marques

Faculdade de Ciências da Universidade de Lisboa, Portugal
{jalves,lmarques}@lasige.di.fc.ul.pt, casim@di.fc.ul.pt

Abstract. This position paper describes initial efforts and ideas for the development of a middleware framework to support the operation of adaptive Wireless Sensor Networks applications with real-time and dependability requirements. We identify a set of underlying services that need to be implemented as part of this framework, explaining why they are needed and what they provide. In order to illustrate how this middleware can be used and its potential benefits, we consider the well-known LQER routing protocol to show how it must be changed to incorporate probabilistic real-time requirements and meet them in a dependable way.

Keywords: Wireless sensor networks, dependability, timeliness, middleware.

1 Introduction

Wireless sensor networks (WSNs) have a number of unique capabilities which make them suitable for many different classes of applications. This includes weather monitoring, industrial monitoring and quality control, object tracking and medical monitoring. But the nice features of WSN, like small size, reduced price and ease of use and deployment in the field, also pose a set of challenges to the application/system developer. WSNs have limited power supply, limited memory and processing power, and may need to operate under harsh conditions on unpredictable environments, making them more susceptible to external faults. The primary focus of the research community has been on addressing these challenges, designing protocols and techniques which try to be as energy efficient as possible, and also robust against the potential faults.

A problem that has been far less considered is that of achieving dependable solutions. Dependability can be defined as "the ability to deliver a service that can justifiably be trusted" [2], and can be broken down into six attributes: availability, reliability, safety, confidentiality, integrity and maintainability. These attributes, and dependability in general, are particularly important in WSN applications involving the control of critical infrastructures, real-time coordination of robots or other vehicles, or involving life-care services.

* This work was partially supported by the EU through the KARYON project (FP7-288195) and the FCT through the Multiannual Funding Program.

M. Vieira and J.C. Cunha (Eds.): EWDC 2013, LNCS 7869, pp. 16–23, 2013.

Since applications using sensor networks rely on the availability and timeliness of the collected sensor data, one way of looking at dependability in this context is to reason in terms of these attributes. In fact, timeliness is not by itself an attribute of dependability, but instead the probability that timeliness requirements are met (i.e., that the system is reliable with respect to timing faults). In [5] the authors explore the concept of perception quality, and observe that as time passes, the quality of our perception of a sensed environment variable diminishes. In other words, the similarity between the real observed event and its systems internal representation tends to grow weaker the longer the sensing took place. As such, a dependable network is understood as one that is able to deliver the sensed data within a temporal bound, and do that with a desired probability.

In this paper we provide initial ideas for a middleware aimed at supporting dependable applications and services with timeliness requirements, running over wireless sensor networks. Achieving hard-real time properties in WSNs is an unrealistic objective [10]. Therefore, our work is intended to support adaptive services and applications, which are able to adjust their behavior at run-time to meet available resources and environment conditions. Dependability objectives are achieved through the adaptation process rather than by securing some fixed temporal bound. In practice, assumed bounds will be secured with a given probability, and the middleware will bring awareness about the relation between bounds and the probability of securing them at any given moment. The resulting programming model will be adequate to develop these applications in environments that exhibit uncertain temporal bounds with some degree of stability.

The paper is organized as follows. In Section 2 we go over related work on the topics of dependability and timeliness in WSNs. Then, Section 3 addresses the middleware design, underlying assumptions, goals and architecture. In Section 4 we provide a brief overview of how the middleware can be used to address timeliness requirements, considering a routing protocol as a toy example. Section 5 concludes the paper and refers to future work.

2 Related Work

2.1 Dependability in WSNs

A lot of research has been done on solving particular issues inherent to wireless sensor networks, such as energy conservation [1] or traffic reduction [8], a topic that has not received as much attention has been that of dependability. The dependability of a system reflects the trust a user has in that system, i.e., that the system will operate as expected [2]. Although dependability has several attributes, availability and reliability are the more relevant ones to our focus on timeliness. For a system to be highly available it means that its downtime is very small (which is usually expressed as a percentage of uptime in a given time frame). Reliability stands for the probability that in a time interval [0,t] the system will operate properly and continuously. In the context of wireless sensor networks, a dependable, reliable network, is a network where for a long sensing

period the data is sensed, sent and delivered to the destination with a very small (or no) downtime and with a very small loss of data [12]. When considering also timeliness requirements, as we do, timing faults must be avoided, and this need must be reflected in the notion of dependability.

In [12], the authors propose an event-based middleware service to improve the dependability of the network. This service uses the publish-subscribe scheme, and divides the network into clusters, where the cluster-heads serve as event brokers to the remaining nodes in a cluster (the publishers, or event sources) and to the base-station, which acts as subscriber. When a cluster-head fails, a network reconfiguration phase is triggered and the other cluster-heads have the responsibility of "adopting" the nodes of the failed cluster.

In contrast with the previous example, we want to provide support for awareness of the timeliness of the network operation, as a means to allow applications to adapt when some nodes fail or stop responding and when the communication latency varies over time (e.g., due to interferences, mobility, or other factors). We thus provide enriched dependability guarantees.

2.2 Timeliness in WSN

Timeliness has been often overlooked in detriment of energy consumption. But as WSNs delve evermore into the realm of time-sensitive applications and scenarios, it becomes increasingly important to address timeliness requirements in the context of these networks. One issue that must be addressed is the very definition of timeliness. In classical real-time computing, timeliness usually stands for the execution of actions in a bounded and fixed, well-known, amount of time. However, strictly satisfying deadlines requires a set of assumptions and system models that do not hold for WSNs [9]. In WSNs, by their very nature, guarantees about communication latency cannot be given in a strict sense. The network does not exhibit deterministic behavior due to the open environment and resources sharing. Communication is subject to varying and uncertain message delays and loss, which makes worst case analysis very difficult or even impossible [9]. Thus, the singular constraints of WSNs require a different characterization for the notion of timeliness.

In [9], a generalized notion of timeliness is introduced, which takes into account the particular nature of WSNs. This notion is built on top of the idea that applications should not request infeasible degrees of performance from the network (a request for a strict deadline on an individual message, is an example of such a request, mainly because of the uncertainty associated with end to end latency in WSNs). The authors define formally, that the generalized notion of timeliness is composed by a time interval, which delimits the target end-to-end transmission interval for a sequence of messages, and by a level of confidence, the probability of successful end-to-end transmissions within the time interval.

We have also previously exploited the idea of providing probabilistic guarantees in alternative to strict hard real-time guarantees [6]. We argue that no matter the approach used to make WSNs more predictable, perturbations are still likely to occur, and as such, instead of specifying fixed upper bounds on

system variables (latency, maximum number of omissions, etc.) monitoring and adaptation techniques should be used to characterize and deal with the uncertainty. Therefore, we proposed a technique based on non-parametric statistics, a lightweight method of statistical inference, which can be used to characterize environment conditions or, in other words, the state of the network. This information can then be used for adaptation purposes. Our current work exploits the availability of such monitoring service, including it in a middleware layer designed to support the development of adaptive real-time applications on WSNs.

3 Middleware Description

The middleware is designed with the following two principles in mind:

Principle 1: Realistic assumptions about WSNs technology and deployment.
Principle 2: Probabilistic timeliness as opposed to hard real-time guarantees.

One of the problems with previous work done on the topic of timeliness and real-time in WSNs has been that it is based on naive assumptions that severely restrict their applicability to real-world systems [10]. Following Principle 1 we will stay clear of such assumptions, whether they are about network and environment conditions, the goals of the architecture, or the applications using it.

Another reason to stay clear of restrictive assumptions is to allow the middleware to be applicable to a broader range of environments. The middleware design should, as much as possible, be independent of the underlying network topology and dynamics and of the expected applications and services running on top of it. Lets consider, for example, routing protocols as potential services running on top of the middleware. Both a flat, multi-route protocol, and an hierarchical cluster-based protocol should be able to take advantage of the middleware although it was not designed specifically for either of them.

One assumption about the environment that is necessary, is that its behaviour, while uncertain, has limited dynamics, i.e. the environment does not change to rapidly in relation to the perception capabilities of the system. The results in [7] validate this assumption, showing that adaptation can be done which is very close to the theoretical perfect adaptation. This implies that the environment dynamics are limited, otherwise this match would not occur, since the bound values would be very different by the time the adaptation was complete.

As stated earlier, achieving hard real-time guarantees is an unrealistic goal for wireless sensor networks. Therefore, in accordance with Principle 2, the design takes from the work in [6] and [9] and exploits the notion of probabilistic timeliness guarantees.

With these principles in mind, we propose a middleware design that can be described as a set of components or services that should run at each node in the network and that provide other applications and services with useful information about the runtime state of the network to support adaptation

The main component is the monitoring service. This service is responsible for monitoring system metrics, such as communication latency or node connectivity, node energy or packet loss rate. The monitoring service itself can be designed as a set of components, one for each monitored metric, and a communication interface. The other components are in essence auxiliary to the monitoring component, providing services it needs to accomplish its goal. These components are the clock service and the non-parametric statistics service.

The clock service is explicitly included as an independent service, as it constitutes a fundamental abstraction for distributed monitoring of time intervals or, in this case, of communication latencies. Without a notion of global time it would be impossible for the monitoring system to perform measurements of end-to-end latencies, as required to provided the intended service. We are aware that techniques based on round-trip delay measurements can be used to measure communication latencies, which could lead to the idea that no global notion of time is needed. However, these techniques requiring message exchanges are just implicit forms of clock synchronization, even if no clocks are explicitly synchronized. On the other hand, a clock service providing global time can be implemented without the need to exchange messages, therefore standing as a building block on its own. For instance, it might be possible that nodes have GPS-synchronized clocks, or that they use the power lines for clock synchronization. If these methods are not available then a clock synchronization algorithm, such as the one in [11] could be used.

The non-parametric statistics service [6] is the component responsible for realizing the statistical operations over sample data provided at its input interface. The monitoring service collects the required sample data, that is, measurements of message delays for messages exchanged with neighbor nodes, and feeds the statistics service to obtain the probabilistic distribution for these message delays. Based on that, it will be possible to answer questions like "what is the probability that a message will be delivered within X time units?". Of course, it may be possible to disseminate latency measurements to other nodes and hence feed the statistics service with the necessary data to raise awareness about the (probabilistic) latency to any other node in the network. In many cases, namely if there is a sink node in the network, the relevant latency measurements will concern the communication between nodes and the sink.

The monitoring component we have just described will be accessed by other services or applications through an interface. As an example, consider the interface for end-to-end latency estimates:

```
(latency l) <- getLatency (node n, probability p)

(probability p) <- getProbability (node n, latency l)
```

These two simple functions provide the basic service of the monitoring component. In the first function a node can inquire the monitoring service about the latency between itself and a node n that will hold with a given probability p. For example, a sensor node that is sending temperature measurements to a sink

node can become aware of the the latency bound that will hold with a certain probability, say, 0.98. If this latency is too high (which can imply that the sink may sometimes have a temporally inconsistent view of the temperature), then maybe the node will decide to increase the frequency of updates in order avoid such temporal inconsistencies with the indicated probability (0.98).

The second function is the complement of the first. In this case the application will inquire the middleware about the probability that a given latency l to a node n will hold. For example, the same temperature sensor node may want to know how probable it is that the latency to the sink will be, at most, 200ms. If the probability happens to be very low, then this may trigger some reconfiguration of adaptation of the node behavior in order to achieve a more predictable behavior.

A similar style of interface can be designed for other monitored metrics, and more powerful constructs can be built, in addition to these operations and maybe using them. The tradeoff between providing a richer middleware is the overhead in terms of needed resources, which are scarce in WSNs. This is why a non-parametric approach, which is light-weight, is considered for the statistics service.

4 Sample Application

In the following paragraphs we provide an example of how the support middleware could be used to enhance an existing service, by allowing this service to perform in a probabilistic real-time way, instead of being simply best-effort.

We consider a routing service, and focus on the modification of the LQER (Link Quality Estimation based Routing for Wireless Sensor Networks) protocol [3]. LQER is inspired by MCR (Minimum Cost Routing) [13] and MHFR (Minimum Hop Field based Routing), and utilizes the concept of a dynamic sliding window of length k for storing historical data of link quality, i.e., the success or failure of the last k transmissions on a given link.

The LQER protocol starts with the minimum hop field establishment, which has the goal of setting up the optimal path to send data to the sink, for each node. In this stage the sink broadcasts an ADV (advertisement) message which contains the hop count to the sink (0 at the sink). This message will be propagated through the network using the flooding algorithm and when a node n receives an ADV message from a node m, it will compare its hop count (h_n) to the advertised hop count of node m (h_m). If $h_m + 1$ is smaller than h_n then h_n is set to $h_m + 1$ and n broadcasts the ADV message with hop count equal to h_n. If $h_m + 1$ is equal to h_n then n adds m to its forwarding table but does not broadcast the ADV message. If $h_m + 1$ is bigger than h_n, then n simply ignores the message. At the end of this stage each node should be able to calculate the minimum hop count to the sink and have a forwarding node set.

Once the minimum hop field for each node has been established, nodes can start routing messages to the sink. When a node needs to send a message to the sink it will choose the node from its forwarding table with the best link quality, that is the node which has the largest value of $\frac{m}{k}$ where m is the number of

successful transmissions among the last k transmissions in the sliding window for that link. After the message is forwarded, the sliding window is updated to account for the success or failure of that transmission.

LQER uses loss as a link quality metric, but the use of the support middleware allows us to consider other time related metrics, such as the probability of success to meet a certain deadline (expressed as the end-to-end latency to the sink) or the estimated end-to-end latency for a desired probability of success. The necessary adaptations for the protocol to operate with these metrics, take place in the Link Quality Table Maintenance Algorithm and the Link Quality Estimation Routing Algorithm. Bellow we provide an overview of the baseline algorithms and of the necessary adaptations for each of the two discussed metrics to be used.

The majority of the changes happens in the Link Quality Table Maintenance Algorithm. Instead of storing the success or failure of a given transmission on some link, the algorithm first queries the middleware (calls the function getProbability) for the probability of sending data through that link within some deadline (which may be a configuration parameter of the routing protocol). Then, it stores the returned value, by first deleting the oldest value on the table and inserting the new one. When choosing a node from the forwarding table, the algorithm will choose the node with the highest average of the probability values stored in the Link Quality Table.

The changes necessary to use the second metric are very similar to the above, but instead of querying the middleware for the probability of meeting a deadline, the algorithm provides a probability and receives from the middleware an estimate of the best possible end-to-end latency for that probability (calls getLatency). When choosing a node from the forwarding table, the algorithm will choose the node with the smallest average of the end-to-end latency values in the Link Quality Table.

5 Conclusions

In this paper we proposed a support middleware for applications with dependability and timeliness requirements. We defined the middleware as a set of components, a main monitoring service and a suite of auxiliary components, which provide, through a programming interface, useful functions to applications running on WSNs. We showed what those functions might look like and provided an example to illustrate how they may be used to in a protocol that takes into account probabilistic timeliness requirements instead of simply exhibiting best-effort behavior.

Our future work will focus on implementing the proposed middleware and an example application, so as to study and evaluate the achievable benefits in terms of the capability to effectively support adaptive real-time applications. We intend to measure the overhead introduced by the middleware, in comparison to a baseline implementation providing best-effort service.

References

1. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: A survey. Ad Hoc Netw. 7(3), 537–568 (2009)
2. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing 1(1), 11–33 (2004)
3. Chen, J., Lin, R., Li, Y., Sun, Y.: Lqer: A link quality estimation based routing for wireless sensor networks. Sensors 8(2), 1025–1038 (2008)
4. Garces-Erice, L., Rooney, S.: Dependable actuation in wireless sensor networks. In: 2007 IEEE Globecom Workshops, pp. 1–5 (November 2007)
5. Marques, L., Casimiro, A.: Data validity and dependable perception in networked sensor-based systems. In: 2010 29th IEEE Symposium on Reliable Distributed Systems, October 31-November 3, pp. 358–362 (2010)
6. Marques, L., Casimiro, A.: Lightweight dependable adaptation for wireless sensor networks. In: 4th International Workshop on Dependable Network Computing and Mobile Systems (DNCMS 2011), Proceedings of the 30th IEEE International Symposium on Reliable Distributed Systems Workshops, Madrid, Spain, pp. 26–35 (October 2011)
7. Marques, L., Casimiro, A.: Evaluating lightweight dependable adaptation in 802.15.4 wireless sensor networks. Technical Report 2012;04, Faculdade de Ciências da Universidade de Lisboa (2012)
8. Ngai, E.C.-H., Gelenbe, E., Humber, G.: Information-aware traffic reduction for wireless sensor networks. In: IEEE 34th Conference on Local Computer Networks, LCN 2009, pp. 451–458 (October 2009)
9. Oliver, R.S.: An Approach to Timeliness in Wireless Sensor Network Communications. PhD thesis, Technische Universität Kaiserslautern (September 2010)
10. Oliver, R.S., Fohler, G.: Timeliness in wireless sensor networks: Common misconceptions. In: Proceedings of the 9th International Workshop on Real-Time Networks, RTN 2010, Brussels, Belgium (July 2010)
11. Sichitiu, M.L., Veerarittiphan, C.: Simple, accurate time synchronization for wireless sensor networks. In: 2003 IEEE Wireless Communications and Networking, WCNC 2003, vol. 2, pp. 1266–1273 (2003)
12. Taherkordi, A., Taleghan, M., Sharifi, M.: Dependability considerations in wireless sensor networks applications. Journal of Networks 1(6) (2006)
13. Ye, F., Chen, A., Lu, S., Zhang, L.: A scalable solution to minimum cost forwarding in large sensor networks. In: Proceedings of the Tenth International Conference on Computer Communications and Networks, pp. 304–309 (2001)