Implementing Business Logic on Sensor Nodes

M. Marin-Perianu
University of Twente
Enschede, The Netherlands
m.marinperianu@utwente.nl

T.J. Hofmeijer
Ambient Systems
Enschede, The Netherlands
hofmeijer@ambient-systems.net

P.J.M. Havinga
University of Twente
Enschede, The Netherlands
p.j.m.havinga@utwente.nl

ABSTRACT

Wireless sensor networks (WSNs) will be able to assist industrial and business processes and to render rich functionality in a dependable way. Two key elements that can make this real are: a simple and efficient way of expressing the business logic, and a reliable mechanism for selectively reconfiguring sensor nodes. We present a solution that combines both elements. The main objective is to guarantee the dissemination of business rules to multicast groups of sensor nodes, while striving for energy efficiency and low overhead. Simple cross-layer optimizations are used to achieve this. For scalability reasons, our solution demands only local knowledge, performs local retransmission of lost packets and uses aggregation of acknowledgements. The results of our experimental evaluation indicate a good ability of recovering from serious errors, even under high error rates.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Protocols, Wireless Communications; J.1 [**Administrative Data Processing**]: Business; J.7 [**Computers in other Systems**]: Process control

General Terms

Algorithms, Reliability, Experimentation

Keywords

Wireless Sensor Networks, Reliable Dissemination, Transport Protocols, Business Processes

1. INTRODUCTION

A large range of business processes [2], [10] can benefit from relocating more logic at the point of action, through the use of intelligent sensor networks. To make this real we need reliable and easy to reconfigure sensor nodes that can overcome resource limitations through collaboration. In the following we study this requirement from both perspectives (reconfiguration and reliability) and describe a complete solution targeting a concrete scenario.

The scenario concerns transport and logistics processes [3], which deal with the large-scale distribution of perishable goods (such as flowers) from the producer to the retail shops. Due to human intervention, there are a number of errors to be considered, e.g. goods are improperly stored, lost, loaded incorrectly or delivered to the wrong store. Often during the delivery process, the goods are transported in rolling containers, or Returnable Transport Items (RTIs). The RTIs intended for a specific shop are grouped together, either at the distribution center (in an area called "expedition floor") or in the trailer. By outfitting the RTIs with wireless sensor nodes, errors can be detected and avoided. The tasks of the sensor nodes are: (1) observe the current situation (e.g. climatic conditions, location, time, neighbourhood, etc.), (2) check if the situation matches a set of rules (further referred as *business rules*) specified for each shop, and (3) trigger alarms or even take actions if the rules are not fulfilled.

Due to the dynamics of the business process, the rules are expected to change accordingly. The central system or a mobile operator has to reconfigure frequently the various groups of nodes in a reliable way (see Figure 1). It is therefore desired to have a *multicast reliable* protocol, which disseminates the new rules to the specified *group* and affects as less as possible the other nodes (in terms of energy and bandwidth).

The main requirement of such a protocol is to correctly deliver *all* the data to *every* intended recipient. However, in the context of WSNs, this requirement exhibits unique challenges due to the scarcity of available energy, memory, computing power and bandwidth. To answer to these challenges, we propose a compact form of expressing and executing the business logic through rules, and an energy efficient, reliable dissemination protocol. We perform a practical evaluation of the integrated solution and draw conclusions based on the experimental results.

2. BUSINESS RULES

The business rules represent a simple, yet powerful method of programming sensor nodes. The structure of the rules aims to express the business logic in a compact and efficient way. Since the sensor nodes will assist real-life processes, the most common tasks are expected to address the monitoring of parameters against certain conditions. If the conditions are not met or erroneous situations are detected, the nodes should inform the backend system and take corre-

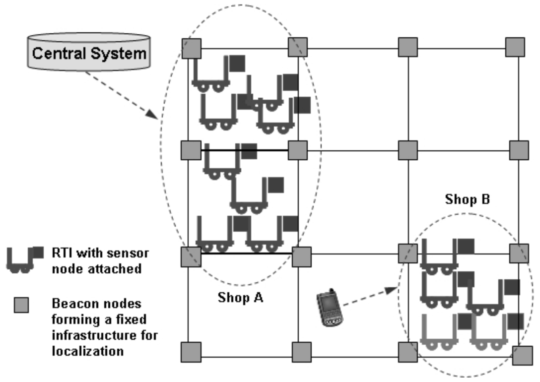

Central System
Shop B
RTI with sensor node attached
Beacon nodes forming a fixed infrastructure for localization
Shop A

Figure 1: The expedition floor.

sponding actions. A simple example would be the following rule: "Measure humidity x at rate r; if it is outside the interval $[x_{min}; x_{max}]$, launch alarm service S_{alarm}". Complex conditions can also be expressed by forming chains of rules (logically linked) and by providing more elaborate actions to be taken when the rules are fulfilled or not.

The structure of a business rule is defined in Table 1. Each rule is evaluated at the specified *sample rate* by testing the values provided by the *sensing driver* against certain *conditions*. The driver usually represents the code that samples the sensed data, but it can be any function that provides a numerical result. In this way, richer functionality, computation or reasoning (e.g. in our scenario, obtain a consensus with the other group members) can be embedded in the rules. The conditions specify interval limits for minimum and maximum values, as well as the admissible variation Δ of the last sample compared with the previous ones. The evaluation of the rule outputs a TRUE or FALSE result that activates a *service*, i.e. executes a user defined code module. Moreover, each rule may be valid only for a certain *running time*, given by the start and stop moments. Finally, in order to construct chains of rules, a *next rule* field is provided.

The evaluation of business rules is carried out by a rule engine, which is a standalone, periodic task. For every rule in turn, according to its sample rate, the rule engine calls the driver function, compares the data against the conditions and launches the appropriate service.

3. TREE-BASED DISSEMINATION

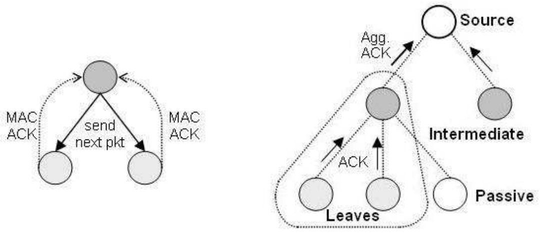
Tree-based reliable multicast protocols (TRMP) [6], [9] are shown to have good scalability properties by delegating responsibility to local groups (i.e., a node and its children in the tree). In addition, a considerable amount of network traffic can be saved by using local and aggregate ACKs. Although WSNs usually rely on a mesh structure, a tree or cluster overlay organisation is often required for a proper, distributed operation. The reliable multicast protocol can thus directly use the available overlay structure. This paper does not focus on the construction or maintenance of such a tree structure, but on its efficient exploitation with respect to the problem of reliable data dissemination.

3.1 Cross-layer approach

In our previous work [8] we reported on experiments with a point-to-point, window-based reliable transport solution. We showed that a cross-layer approach, where the transport layer interacts with routing and MAC, can bring about significant benefits in terms of energy and throughput. One specifically important optimization was to use local MAC ACKs, which have no additional energy cost, and resort to transport layer ACKs only for the whole window. In this paper, we extend the idea for the multicast situation, following the scheme described by Levine et. al [7]: the local ACKs are MAC-based and the aggregate ACKs correspond to window ACKs. In this way we ensure a relatively fast sending rate (because MAC ACKs take no additional time) and perform error detection and repairs locally. The aggregate ACKs are needed for two reasons: (1) MAC ACKs increase the confidence in the correct reception of a packet, but do not guarantee it, and (2) the source must have an indication for safely releasing the data from the memory.

ID	Sensing driver	Sample rate	Conditions (min, max, Δ)	Services	Running time	Next rule

Table 1: Business rule structure



(a) (b)

Figure 2: Protocol operation.

3.2 Protocol Description

We consider the case of a dense network of heterogeneous nodes, organised into multicast groups according to application specific criteria (for example, nodes with similar resources or placed in a certain area). For simplicity, we consider that the protocol operates on a multicast tree rooted at the *source*. The nodes of the tree are either *members* of the group or *forwarders* needed for propagating the information.

The message to be disseminated is split in fixed size windows of packets. The packets are identified by sequence numbers. The receivers acknowledge the window with ACK or NACK packets indicating through a bitmask the correctly received and missed packets, respectively.

The protocol starts with an initial phase, during which a packet announcing the new message is flooded into the tree and the nodes initialize the dissemination session as follows:

- Set the protocol parameters, such as sequence numbers, message length, etc.
- Forward the message announcement to all children,
- Based on the replies from the children, compute the height in the tree, the timeouts and decide the role,
- Acknowledge the parent and piggyback the height and role.

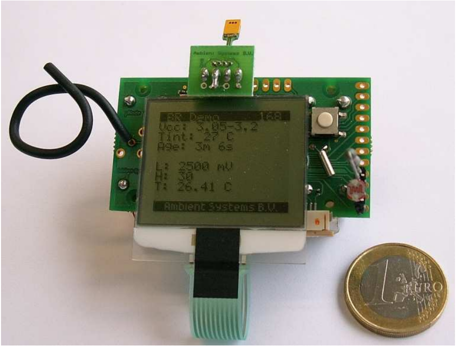


Figure 3: Node with SHT and LDR sensors.

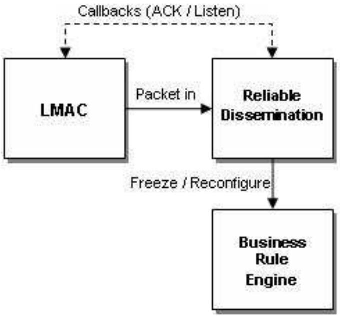


Figure 4: Cross-layer interactions.

Acknowledgements and retransmissions are used to ensure that the initial phase ends up correctly and all interested nodes are ready to receive the message.

After the completion of the initial phase, the source starts to send the message. Each packet from a window is sent until all the direct children acknowledge it through MAC ACKs (see Figure 2(a)) or a maximum number of errors is exceeded. At the end of one window, the source waits for transport level ACKs/NACKs from all the direct children. If there are NACKs or a timeout occurs, the source will start resending the missed packets (or the whole window), following the same procedure. Accordingly, the leaves of the tree (i.e., nodes that do not have children that are members or forwarders) will acknowledge every packet at MAC level and every window at transport level. The intermediate nodes carry out a double task, being both parents and children, and therefore they are the most exposed in terms of energy consumption. An intermediate node maintains a small cache, equal to the window size, in order to act as a source for its children, taking care of all local repairs. Only after all the children have sent window ACKs, it can send an aggregate window ACK to its parent (see Figure 2(b)).

3.3 Errors

Our protocol can recover from the following types of errors:

- *Packet losses.* Packets may be lost or corrupted when transmitted from parent to children. These errors are detected by the parent at MAC layer or signaled by the children through window NACKs.

- *Faults.* By fault we mean an error that causes the retransmission of the entire current window. There are two possible reasons for a fault: a topology change (a node loses the contact with its parent and has to register to another parent) or a hardware error (due to imperfect battery contacts, harsh environments, watchdog behaviour, etc.).

4. PRACTICAL EVALUATION

In this section we present the implementation of the complete solution for reliably disseminating business rules to groups of sensor nodes and we describe the tests performed within an experimental setting.

4.1 Platform

In our experiments, we are using Ambient uNode 2.0 platform [1] (see also Figure 3). The onboard micro-controller is the Texas Instruments MSP430F1611, which offers 48kB of Flash memory and 10kB of RAM. The radio transceiver has a maximum data rate of 100kbps. The uNodes run AmbientRT [5], which is a real-time multitasking operating system designed for supporting data centric architectures. The key points of AmbientRT are:

- *Real-Time Scheduler*, also providing mutual exclusion for resource sharing;

- *Data Manager*, based on a publish/subscribe mechanism for inter-task communication;

- *Dynamic Loadable Modules (DLMs)*, which support system reconfiguration at runtime.

4.2 Cross-layer interactions

As mentioned in Section 3.1, we rely on a cross-layer design for increased efficiency. Figure 4 sketches the interactions between the main modules: LMAC, the reliable dissemination protocol and the business rule engine. LMAC [4] is a lightweight medium access control protocol that provides the following features:

- Collision avoidance through scheduled access,

- Neighbour information, including neighbour ID, link quality and distance to gateway (in hops),

- Acknowledgments and retransmissions, within the local neighbourhood,

- Control points through callbacks, an extension of the original LMAC that practically enables the cross-layer interaction.

LMAC delivers the received packets and notifies the MAC ACKs to the dissemination module. The latter, in turn, instructs LMAC to listen only to those packets intended to the specific multicast group. Considerable energy is saved by applying this selective listening scheme.

When a new set of rules is announced in the initial phase of the dissemination, the sensor nodes enter a reconfiguration state. Since both the business rule engine and the reliable dissemination are critical tasks and may compete for the communication medium, we prevent them from running

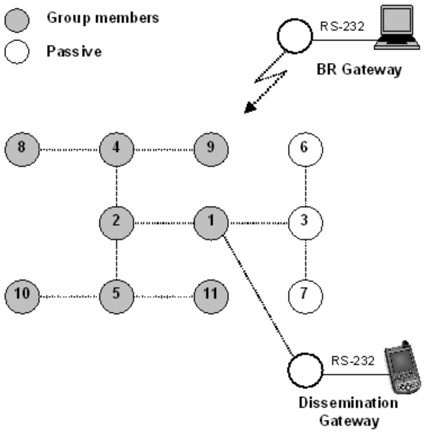


Figure 5: Experimental setting.

ID	Name	Driver	Explanation
1	V_{cc}	internal	Battery level
2	T_{int}	internal	CPU temperature
3	Age	counter	Running time
4	$Light$	LDR	Light level
5	T	SHT	Temperature
6	H	SHT	Humidity
7	$Message$	push-button	User message

Table 2: Example set of business rules.

Parameter	Value
Packet size	32 bytes
Window size	5 packets
MAC frame length	1 sec.
Tree height (max. hops)	4
Tree degree (avg. children)	2

Table 3: Experimental parameters.

in parallel. The dissemination protocol uses exclusively the radio to transfer the data, while the rule engine is stopped. However, in practice, we have to account for unexpected errors in the dissemination process. Failures of the dissemination should not corrupt the entire operation of the nodes. For this, as long as the dissemination process is still active, it "freezes" the rule engine task by periodically postponing its activation (similar to a watchdog behaviour). If an exception occurs and the dissemination does not terminate correctly, the business rule engine eventually resumes its execution. If the new rules are safely transferred, the dissemination process forwards them to the rule engine for reconfiguration. Both partial and complete reconfiguration are supported: new rules can be added to the current set or can replace previous rules with the same IDs.

4.3 Tests

We used for evaluation a multihop network of 11 nodes placed as indicated in Figure 5. The group members correspond in our scenario to the RTIs for a given shop. We used the dissemination protocol to reliably deploy and change sets of rules for the group of nodes. A typical set of rules is presented in Table 2. Besides the internal voltage and temperature sensors, the nodes were equipped with the following: one light dependent resistor (LDR), one temperature and humidity sensor (SHT) and one push-button (see rules 4-7).

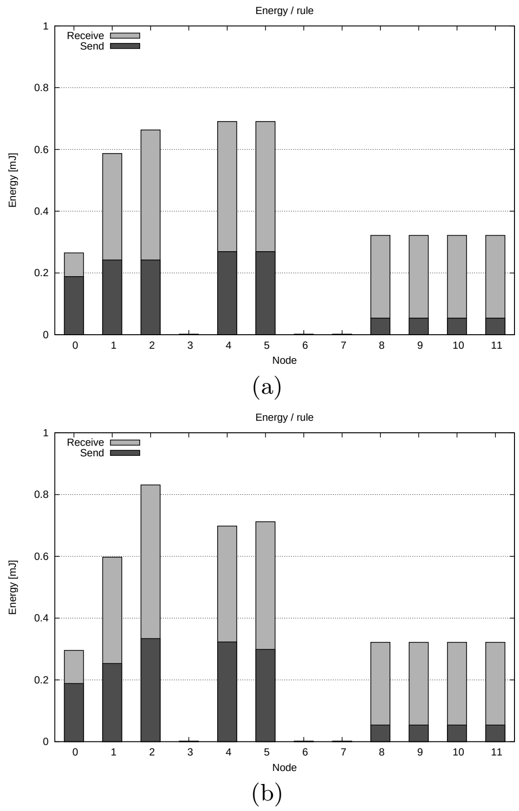
We used for dissemination a mobile gateway composed of a sensor node connected to an iPAQ through serial interface. A second gateway (business rules gateway) was logging the results issued by the business rules running on the nodes. The communication on the serial links was done reliably by using a simple stop-and-wait ARQ protocol. For efficiency reasons, the packet size was chosen to fit one business rule. We performed 30 experiments, in each disseminating a set of rules to the multicast group from Figure 5. In all the cases, the nodes reconfigured correctly. The total disseminated data (over all experiments) cumulated $\approx$ 4kB. Each node stored logging information (such as the number of packets sent and received, the type and number of errors, etc.) and reported it back to the dissemination gateway. The user could thus follow on the iPAQ the ongoing process and an estimation of the energy consumption in the network.

We now discuss the most important numerical results. The observed overall packet loss rate was 2,4%. Moreover, all lost packets have been received properly at the first retransmission. This result points out that analysing MAC ACKs and performing local retransmissions represent a good choice. The average time for transferring one business rule was 2.7 seconds, but this value depends substantially on the parameters of the setting, listed in Table 3. Figure 6 (a) gives an overview of the average energy spent by each node for disseminating one rule. Node 0 represents the source, i.e. the dissemination gateway. As expected, the intermediate nodes (nodes 1, 2, 4 and 5) are the most loaded, being involved in forwarding both data and acknowledgements. The amount of energy spent on transmissions and receptions is balanced. In contrast, the leaf nodes and the source consume energy mostly on one operation (sending or receiving), according to their role. Finally, the nodes that are not group members (nodes 3, 6 and 7) remain completly passive during dissemination and save energy.

We performed a second round of 30 experiments, in order to test the behaviour of our system under higher error rates. We had to provoke the errors ourselves by randomly resetting nodes involved in dissemination and forcing in this way the protocol to recover from faults. The ratio of faults was on average 1.5%. This had also a marginal effect on the packet loss rate, which increased at 3.3%. It took, however, at most two retransmissions to repair a packet loss. The most affected parameter was the average time per business rule, which raised at 6 seconds. Figure 6 (b) shows still reasonable values for the energy consumption; the increase computed over the whole network is $\approx$ 5%. This proves that the system has good ability of recovering from serious errors, even under high error rates.

5. CONCLUSIONS

Our work focuses on integrating WSNs into real-world business applications. We approach this from two directions: reconfiguration and reliability. We propose a simple, yet efficient method of expressing the business logic, by means of rules through which the nodes can verify if the observed situations correspond to the proper conditions. Due to the dynamics of the real-world processes, the rules are expected to change often. Therefore, we devise and test a



(a)

(b)

Figure 6: Energy consumption.

reliable dissemination protocol addressing multicast groups of nodes. We argue that such an approach is more selective and scalable than unicast-based or flooding and, at the same time, maps better to the application scenarios.

Our experimental results show the feasibility of a resource-lean dissemination layer, which interacts with lower layers, such as MAC, for increased efficiency. Nevertheless, the implementation work on the sensor nodes revealed that the cross-layer approach is indeed useful only if the interaction methods between layers are enough uniform and general. Otherwise, the code becomes error prone and difficult to maintain or reuse.

We finally list several brief observations occurring from our experiments. First, simple techniques, relying only on local knowledge, should be favoured, as they can bring about the desired reliability while leaving more resources for the application space. Second, all theoretical assumptions about link errors or node failures should be carefully considered. In practice, all errors eventually happen and most usually in a burst manner. Third, parameters such as timeouts, window size or retry bounds prove important and their values may affect considerably the real behaviour of the protocol. The best values, however, can be determined only experimentally and do not match all situations. Therefore, we consider improving our protocol with an adaptive behaviour for future work.

6. ACKNOWLEDGMENTS

The authors would like to thank Lodewijk van Hoesel, Stefan Dulman and Nirvana Meratnia for their constructive comments. This work has been partially sponsored by the European Commission as part of the CoBIs project (IST 004270).

7. REFERENCES

[1] Ambient System. http://www.ambient-systems.net.

[2] Collaborative Business Items (CoBIs). http://www.cobis-online.de.

[3] L. Evers, M. J. J. Bijl, M. Marin-Perianu, R. Marin-Perianu, and P. J. M. Havinga. Wireless sensor networks and beyond: A case study on transport and logistics. In *International Workshop on Wireless Ad-Hoc Networks (IWWAN 2005)*, 2005.

[4] Lodewijk Van Hoesel, Tim Nieberg, Jian Wu, and Paul Havinga. Prolonging the lifetime of wireless sensor networks by cross-layer interaction. *IEEE Wireless Communication Magazine*, 12 2004.

[5] T. Hofmeijer, S. Dulman, P. G. Jansen, and P. J. M. Havinga. AmbientRT - real time system software support for data centric sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 61–66. IEEE Computer Society Press, 2004.

[6] Brian Neil Levine and J.J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Syst.*, 6:334–348, 1998.

[7] Brian Neil Levine, David B. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In *MULTIMEDIA '96: Proceedings of the Fourth ACM International Conference on Multimedia*, pages 365–376, New York, NY, USA, 1996. ACM Press.

[8] Mihai Marin-Perianu and Paul Havinga. Experiments with reliable data delivery in wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing Conference (ISSNIP)*, pages 109–114, Melbourne, Australia, December 2005. IEEE Computer Society Press.

[9] K. Obraczka. Multicast transport protocols: A survey and taxonomy. *IEEE Communications Magazine*, 36(1):94–102, 1998.

[10] Martin Strohbach, Hans-Werner Gellersen, Gerd Kortuem, and Christian Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *Ubicomp*, pages 250–267, 2004.