RECOUP: Efficient Reconfiguration for Wireless Sensor Networks

S. Pennington, A. Waller, T. Baugé
Thales Research and Technology,
Worton Drive, Worton Grange, Reading, RG2 0SB
Sarah.Pennington@thalesgroup.com

ABSTRACT
In this paper, we describe RECOUP (Reliable Configuration Update), an efficient protocol for updating the configuration of a Wireless Sensor Network (WSN).

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols.

General Terms
Management, Reliability

Keywords
Wireless sensor networks, protocols, configuration management

1. INTRODUCTION
Within a WSN, a management framework may be needed to allow configuration parameters to be changed after the initial deployment of the network. For example, the level of security for sending and receiving messages may need to be changed. The security level can be increased to provide higher protection against attacks from malicious sensor nodes and it can also be decreased again when the threat subsides to conserve power. In order to implement this framework, all sensor nodes in the network need to be instructed how to set their configuration. This needs to be done in a reliable, secure, synchronised, and efficient way.

The RECOUP protocol ensures that all nodes in a network receive configuration management messages that inform them to update their configuration. It ensures that all nodes have a consistent configuration, and allows recovery of situations where some nodes are in an inconsistent state. It is particularly suited to applications that send small configuration payloads, so that the configuration is contained within a single packet. In addition, the protocol is appropriate for applications where speed of update and power consumption is of importance, or where nodes need to recover from an inconsistent state, for example new nodes joining the network.

2. RELATED WORK
Protocols to achieve reliable reconfiguration in wired networks exist, such as reliable IP multicast. However, these are not appropriate for WSNs due to their significant overheads in terms of bandwidth used and maintenance of state on network nodes. Moreover, these approaches do not inherently allow recovery from an inconsistent network state, such as may occur when a new node joins.

Reconfiguration taking into account the significant constraints of WSNs is a relatively new area of research for which few solutions have been proposed. Of those that do exist, most, such as MOAP [1], Deluge [2] and TinyCubus [3] are designed for distributing code updates, which are assumed to be large. Their main aim is to save communications and memory costs where multiple packets of data need to be sent and speed of update is a minor issue.

Many assumptions and optimisations used by these approaches, such as the use of negative acknowledgments from receivers to signal missed packets, are not valid for small, single packet, updates which need to be disseminated rapidly. In addition, use of a periodic broadcast by existing nodes of the current code version to allow new nodes or nodes that have been out of range to update themselves is relatively inefficient and will lead to significant delays in (re-) incorporating such nodes in the network.

Finally, synchronisation of the updates is rarely considered. It is generally assumed that the network will be unable to perform its usual operations (e.g. collecting sensor data) while the code update is being distributed. For code updates, this is not a problem as these will be relatively infrequent. However, for frequent security level changes for example, this would be a significant issue. In [4], small configuration updates are considered, but the use of TCP/IP is suggested for point-to-point updates, which is inefficient, and reliable broadcast updates are left as future work.

3. RECOUP
The protocol has two major features. Firstly, new configuration updates will be flooded throughout the sensor network using a smart flood mechanism. Secondly, if nodes have missed updates, e.g. due to being out of range when the last update was sent or because they are new nodes being added to the network, then the protocol provides a 'local repair' (update) mechanism, which is triggered on the next transmission from the out-of-date sensor. This provides a rapid repair mechanism should nodes have inconsistent configurations and ensures that the network has a very high probability of being in a consistent state. Note that the protocol makes no assumption as to the communication pattern of the sensor network (peer to peer, tree structure, etc), or the network's topology (single or multi-hop) and is patent pending.

For the following description, we assume, without loss of generality, a gateway sensor node that is connected to a PC running the management application and a network of sensor nodes. When the configuration of the sensors needs to be changed, the management application on the PC will send a

MobiQuitous 2008, July 21-25, 2008, Dublin, Ireland.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.4049
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.4049

request to the gateway node. The gateway will then send out an update to all the sensor nodes in its immediate neighbourhood using a link-level broadcast of a 'configuration management message' that contains this new configuration. To enable nodes to tell which configurations are more recent, version numbers or time stamps can be used.

When a sensor node first joins the network it will update to the first valid configuration it receives. When the new node receives a sensor data message from another node, it will broadcast an invalid configuration. This alerts other nodes in the network that it is out-of-date and nodes with a valid configuration will broadcast their configuration. On receiving this valid configuration, the new node will immediately update to this configuration.

When a sensor node that has a valid configuration receives a configuration management message, if the received message is more recent than the previous message the node received then it will immediately change its configuration. It then broadcasts this new configuration in a configuration management message to its neighbours. In this way, the update will be flooded throughout the network. If the received message is less recent, it will broadcast its own, more recent, configuration to its neighbours. However, if the versions are the same, it will simply ignore the message, preventing the message from being flooded indefinitely.

A local repair is achieved as follows. When a sensor node receives any message that is not a configuration management message, it first checks that the configuration of the node that sent the message is the same as its own configuration (note that the protocol requires that the configuration of the sending node can be unambiguously determined from the received message). If the two configurations match, then the message is processed normally. If they do not match, then the node that received the message may either have a more recent or older configuration. To determine which is the case, it sends its own configuration to its neighbours. As described above, on receipt of this message either the other nodes will update their configuration (if they have older versions) or will respond with the newer configuration (if they have newer versions).

In the case that on receiving a message a node detects a mismatch in configurations, the received message may either be processed normally or dropped, depending on the requirements of the application. For example, when sending security management messages using this protocol, if the current security policy requires packets to be sent with authentication but the received message is unauthenticated, then the data should be regarded as potentially compromised and the packet should be dropped by the node. However, if the received message were authenticated, then regardless of the current security policy any compromise of the data would be detectable. In this case, the packet could be processed normally. In this way, the amount of application data lost due to lack of synchronisation of configurations is kept to a minimum.

4. EVALUATION

The full RECOUP protocol and a flooding algorithm (based on the protocol provided with TinyOS, and set to broadcast updates eight times) were implemented on the TinyOS v1.1.15 platform [5]. TOSSIM, a bit-level simulator designed for the TinyOS platform [6], was then used to evaluate the protocol.

We found that the flooding protocol results in eight times the number of configuration update messages being sent compared to the RECOUP protocol. Furthermore, the average packet loss for the RECOUP protocol was lower than that for the flooding protocol. The high number of packets sent using the flooding protocol caused congestion in the network which led to queue overflows and packets being dropped. Both protocols delivered each update to all nodes in the network, although the flooding protocol resulted in a faster update of the network than the RECOUP protocol. However, we demonstrated that the flooding protocol does not guarantee to deliver the update to nodes that join the network after the initial flood. Conversely, the 'local repair' feature of the RECOUP protocol means that it is able to deliver the update to all nodes that miss the initial flood.

5. ACKNOWLEDGMENTS

This paper describes work partially funded by the FP6 EU project "e-SENSE, Capturing Ambient Intelligence for Mobile Communications though Wireless Sensor Networks", Contract Number: IST-4-027227-IP, www.ist-e-sense.org. It also describes work undertaken in the context of the SENSEI project, 'Integrating the Physical with the Digital World of the Network of the Future' (www.sensei-project.eu). SENSEI is a Large Scale Collaborative Project supported by the European 7th Framework Programme, contract number: 215923.

6. REFERENCES

[1] Stathopoulos, T., Heidemann, J., Estrin, D., "A remote code update mechanism for wireless sensor networks", *Technical Report CENS-TR-30*, University of California, L.A. (2003).

[2] Hui, J.W., Culler, D., "The dynamic Behaviour of a data dissemination protocol for network programming at scale", In: *Proc. Of the 2nd Intl. Conf. On Embedded Networked Sensor Systems.* (2004) 81–94.

[3] Pedro José Marrón, Andreas Lachenmann, Daniel Minder, Matthias Gauger, Olga Saukh and Kurt Rothermel, "Management and configuration issues for sensor networks", *International Journal of Network Management -- Special Issue: Wireless Sensor Networks, Volume 15, No. 4,* pages 235—253, 2005.

[4] Markus Anwander, Gerald Wagenknecht, Torsten Braun, "Energy-efficient Management of Heterogeneous Wireless Sensor Networks", *Technical Report iam-07-002*, IAM, University of Bern, 2007.

[5] University of California, Berkley. *TinyOS.* 2005, available at http://www.tinyos.net/

[6] P. Levis et al., TOSSIM: accurate and scalable simulation of entire TinyOS applications, *Proc. of SenSys 2003.*

[7] A. Woo and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of the 1st ACM Conf. on Embedded Networked Sensor Systems*, pages 14--27. Los Angeles, Nov 5-7 2003

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.4049
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.4049