The Design and Implementation of an IP-based Sensor Network for Intrusion Monitoring

Adam Dunkels, Thiemo Voigt
Swedish Institute of Computer Science
{adam,thiemo}@sics.se

Niclas Bergman, Mats Jönsson
Saab Systems AB
{niclas.bergman, mats.joensson}@saabsystems.se

***Abstract*— We present an experimental deployment of an IP-based wireless sensor network that is intended to operate as an intrusion monitoring system. This network is the first actual deployment of a fully IP-based wireless sensor network with small and computationally constrained sensor nodes.**

The intrusion monitoring system detects motion in a building which should be empty. The detected events are transmitted to an external monitoring entity, as well as logged inside the network. The logged events are distributed throughout the network and can be collected with a PDA inside the monitored building.

We have also learned that the software development process is very time consuming unless support for over-the-air reprogramming is implemented, and that the unpredictability of radio conditions make sensor node placement hard.

I. Introduction

Wireless sensor networks have received a lot of attention from the research community during the last years. Nevertheless, most of the work is done in simulation with only a few actual deployments. The deployments that have been made (e.g. [12]) have provided insights that had not been possible to get in a simulation environment.

In this paper, we describe an experimental deployment of a sensor network for intrusion monitoring. This network has been developed by SICS and Saab Systems as part of the DTN/SN project [1]. It was successfully demonstrated at Saab in Järfälla, Sweden, in May 2004. The network serves as an intrusion monitoring system that can be rapidly and temporarily deployed at a location, and easily be transported to another location when the current task is over.

The main task of the sensor network is to detect movements in a building which is supposed to be empty. If movement is detected, the network sends an alarm to a command center that will alert a security team. The security team enters the building carrying a PDA that communicates with the network. The network provides the team with logs of alarms, as well as the continued monitoring of the building.

The hardware for our network consists of ESB boards described in Section II. While the ESBs are equipped with a number of different sensors, our application only uses the motion detector. All nodes run the Contiki operating system [5] (Section III), our lightweight operating system for tiny networked devices.

To make it possible to bridge the sensor network over IP-based channels, such as GPRS or LANs, communication between the sensors within the sensor network is made using TCP/IP. The TCP/IP support is provided by uIP [3], a complete TCP/IP implementation with a code size of only a few kilobytes requiring only few hundred bytes of RAM.

Fig. 1. Hardware platform: ESB from FU Berlin

This network is the first actual deployment of a fully IP-based wireless sensor network made of small and computationally constrained sensor nodes. This work points towards the possibilities with using TCP/IP for wireless sensor networks.

In this paper, we describe the components of our sensor network and present the design and operation of the network. We also report on our experiences in building the network and discuss lessons learned.

II. Hardware platform

We use the ESB sensor board [2], developed by the Freie Universität Berlin (Figure 1) as our hardware platform. The ESB board consists of a Texas Instruments MSP430 low-power micro-controller with 60 kilobytes of flash ROM and 2 kilobytes of RAM, an RF Monolithics TR1001 single-chip RF transceiver, and a collection of sensors. The on-board sensors are:

- A light sensor for the detection of visible light.
- A passive infrared sensor for detection of movement.
- A temperature sensor.
- A vibration sensor for detection of movement of the sensor board.
- A microphone for determination of the ambient noise level.
- An infrared sender and receiver.

The MSP430 micro-controller is designed specifically for low-power applications and provides a set of low-power sleep modes. The micro-controller is awakened from the sleep mode by an interrupt, generated either by one of the internal timers, or by an external device such as the detection of motion by

the passive infrared sensor. The on-chip flash ROM can be selectively reprogrammed by software running on the microcontroller.

III. The Contiki Operating System

The sensor nodes run the Contiki operating system [5] developed at SICS. Contiki is lightweight system for communication-oriented, memory-constrained devices such as tiny sensor devices. Contiki includes our uIP TCP/IP stack [3] and therefore has full TCP/IP support.

In order to ease software development and deployment, Contiki allows individual programs and services to be dynamically loaded and unloaded in a running system. The development of this feature was completed during the development of the network described here, and proved to significantly shorten the time for the development cycle. Additionally, we developed a simple flooding protocol for over-the-air distribution of binary code replacements to a network of nodes. We found that the simple protocol worked well in small networks, but we intend to implement more complex distribution methods (e.g. the Trickle protocol [11]) for larger network deployments.

IV. Scenario

The intended scenario of the network is a peace-keeping or military mission in an urban environment. The sensor network-based intrusion monitoring system is intended as an aid when securing buildings. When securing a building the team places leave-behind sensors in secured rooms and corridors. The purpose of the sensors is to detect future intrusion and threats from, e.g., snipers.

The sensor network is in contact with an external command center, which is notified when there is movement in the building. The command center can then instruct a search team to once again enter the building. The team carries a PDA that connects to the sensor network and downloads information about where movement has been detected. The team may enter the building from any entrance, and it is therefore crucial that the network distributes the detected information throughout the network. Also, if the network should become partitioned because of failing nodes, having the information replicated at many places in the network is important.

V. Network protocols and mechanisms

One of the aims of our network was to make a first test of the viability of using the TCP/IP protocols for wireless sensor networking. The advantage of using IP in the sensor network is the issue of connectivity: with IP running within the sensor network, we can easily connect the sensor network to any other IP network, without protocol converters or proxies [7].

It is often argued that the TCP/IP protocol stack is unsuited for sensor networks because of the specific requirements and the extreme communication conditions that sensor networks exhibit. We believe, however, that there are ways to remedy the problems and therefore that TCP/IP might be a viable alternative for sensor networks [6]: **Spatial IP address assignment** provides semi-unique IP addresses to sensor nodes based on their location. **Application overlay networking** provides data-centric mechanisms, which often are well-suited to the requirements of sensor networks [8]. We have shown that **Distributed TCP caching** significantly increases TCP performance in wireless sensor networks [4]. Finally, we believe that the application-specific nature of sensor networks make **header compression** very efficient.

In our network, we wanted to explore spatial IP address assignment and data-centric overlay routing, as well as the connectivity advantages that we gain by running IP inside the sensor network.

A. IP address assignment

Spatial IP address assignment uses the node's location to construct its address, unlike ordinary IP address assignment where IP addresses are assigned based on the network topology. The location may either be a physical location as in Figure 2 or a logical location (e.g. "next to the door in room A").

Spatial IP address assignment assumes that sensors know their own location. This assumption is valid in many type of sensor networks, as sensor information may be useless unless it can be connected to a physical location. In our network, each sensor is configured with a location as the sensor is deployed.

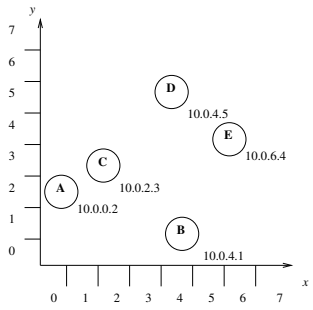

Fig. 2. Spatial IP address assignment

Each node constructs its own address by combining its coordinates with the address prefix for the network. The address prefix of the network is configured during deployment and typically is one of the private IP address spaces [13]. In our networked we used the 172.16.0.0/16 address block with the logical x and y coordinates of the sensor nodes as the last two octets.

Once a sensor node has been given its address, it will be identified only by its location. If two or more sensor nodes are placed at the same location, they will appear as a single sensor.

B. Event propagation using an overlay network

The network (Figure 3) consists of two separate parts: a set of **backbone nodes** make up a backbone network, and a set of **sensor nodes** that send alarm events to the backbone. The backbone network replicates and transports alarm event information so that all backbone nodes will have a log of all recent events from the entire network.

Backbone nodes periodically try to find each other by using broadcast messages. When two backbone nodes have found each other, they open a connection for replicating event logs. Since the radio conditions of the network may change during the network life time, the backbone nodes periodically close and reopen their event replication connections.

Any sensor node can become a backbone node by setting the appropriate configuration parameter. In our network, the decision whether a node would become a sensor node or a backbone node is made during deployment of the network, but we intend to explore semi-automatic methods for future networks.

A security team that enters the network carries a PDA connected to a **mobile backbone node**. The mobile backbone node will connect to nearby backbone nodes and exchange event information using the same mechanism as other backbone nodes.

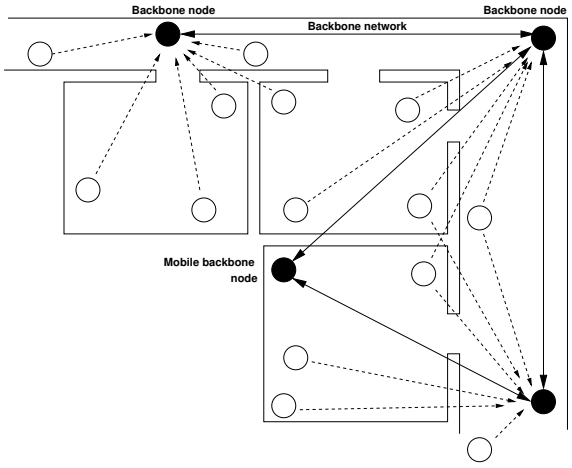

Fig. 3. The network with backbone nodes (black), sensor nodes (white), and the backbone network.

C. Event transmission

When a sensor node detects movement, it broadcasts an alarm notification event together with a time-stamp to all backbone nodes that are in range. The receiving backbone nodes insert the event into their event log and propagate the event on the backbone network. As the alarm event is time-stamped by the sender, the backbone network is able to detect duplicate event notifications and to avoid propagating duplicate events.

Backbone nodes sort their event log by the events' time-stamps. A backbone node will not store an event if a newer event from the same sensor already exists in the log. Also, events that are older than a certain threshold (currently five minutes) will not be propagated. This ensures that old events do not circle around the network.

The sensor nodes periodically transmit heartbeat messages to nearby backbone nodes. Backbone nodes keep track of all sensor nodes seen and the time of their last heartbeat message. The absence of a number of heartbeat messages (currently three) is taken as a sign that the sensor has failed, and the backbone node inserts a failure notification event into the backbone network. If another backbone node has recently seen a heartbeat message from the sensor, it can cancel the failure notification event by inserting a life-sign notification event. This is to guard against a situation where a backbone node flags a sensor as failed, only because the sensor node happened to have a bad radio path to that particular backbone node, while the sensor has perfectly good connectivity with another backbone node.

D. Time synchronization

Because each event is time-stamped, it is important that the entire network has a similar notion of the current time. To synchronize the nodes' time, we use a simple hierarchical scheme where each node periodically broadcasts its own time and maintains a significance level of its time. A node that receives a time message with a higher significance level sets its clock to the time in the message and updates its significance level to one less than that of the received time message. A node also accepts an incoming time message with the same significance level as its own, but only if the time is higher than its own.

Nodes that are connected to a monitoring entity obtains a "correct" time from the external network and therefore have the highest significance level.

E. External network connectivity with NNTP

External network connectivity is important, as alarm events must be sent to a command central. Since the sensor network runs the TCP/IP protocols, external connectivity can be achieved using off-the-shelf technologies such as GPRS. Furthermore, it is desirable to replicate the sensor data outside of the sensor network in order to increase the reliability of the system.

We chose the Network News Transfer Protocol (NNTP) [10] for external connectivity from the sensor network. There is a wealth of software for NNTP replication available, and the purpose of using NNTP for external connectivity is to leverage this existing software infrastructure.

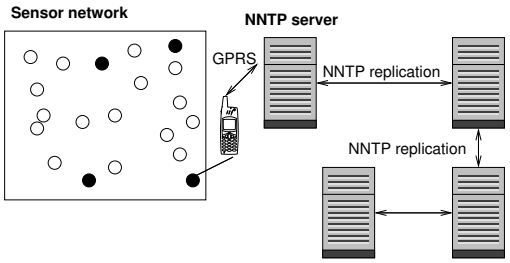

Fig. 4. The sensor network connected to a replicating network of NNTP servers.

All backbone nodes run a small NNTP server which transforms the event log into NNTP messages. The NNTP messages are sent either to a PDA carried by the security team, or to an external NNTP server, from which the data can be further replicated as in Figure 4.

VI. Network deployment and operation

The network is intended to be deployed rapidly, with each sensor node requiring only very little configuration. In our network, each sensor node is manually configured when the node is deployed.

Our prototype sensor boards are equipped with both an infrared receiver and a transmitter which can be used as a short-range communication channel. We utilize the infrared communication to transmit configuration parameters to the sensor nodes. One sensor node is loaded with a configuration that is transmitted to another sensor node by pushing a button on both sensor nodes. This approach have previously been used to transmit keying information for spontaneous networking [9].

We therefore note that the manual configuration step is not a "necessary evil", but that the configuration step is a way to provide the sensor with both configuration parameters, and security information such as cryptographic keys.

Once the network is deployed, it does not require further management.

VII. Experiences and Lessons Learned

Software development and deployment. At the beginning of the project, we had not yet implemented the necessary support for dynamic loading of programs in our Contiki system. Because of this, whenever we needed to make a change to the software, we were forced to manually collect all sensor nodes and manually download new firmware onto each individual sensor node. Downloading firmware onto a sensor node required approximately 40 seconds, and reprogramming the entire network required nearly an hour of work. This lead to a situation where development was slow, and where we would only utilize a small number of the available sensor nodes.

When we had completed the dynamic loading facilities in Contiki, we could do over-the-air reprogramming of the sensor nodes. Since radio is an inherently broadcast medium, we could download a program to a large number of nodes simultaneously. With this facility, the time required to reprogram the network was reduced to one or two minutes; a reduction of an order of magnitude.

Network deployment and radio coverage When deploying and testing our network, we noticed how unpredictable and changing the radio environment was. In general, it was not possible to predict how well a particular sensor node would be able to communicate with another sensor node. Ideally, we would like to have a method to measure the communication environment and present it to the person deploying the network. We tried using the LEDs of our sensor nodes to blink whenever a sensor node heard a message from a backbone node, but this proved to be a too coarse-grained mechanism and did not provide enough information to be able to predict the radio environment.

VIII. Summary

We have developed an IP-based wireless sensor network for intrusion monitoring. The network serves as a first test of a TCP/IP-based wireless sensor network and the spatial IP address assignment mechanism. The network consists of a backbone to which sensors are connected as leaf nodes. Sensors report motion detection events to the backbone network. The backbone replicates the events as a linear event log through the network. The backbone consists of a set of backbone nodes, to which the sensor nodes broadcast events. All events are time-stamped by the sensor nodes and a simple clock protocol keeps nodes' time synchronized.

The network is configured during deployment using the infrared communication capabilities of the sensor nodes. One reason for having manual configuration of each sensor node is that sensitive information such as cryptographic keys can be transmitted together with the configuration information.

We found that software development for a network of sensor nodes is time consuming and that being able to utilize the radio medium for over-the-air programming greatly reduces the development time.

IX. Acknowledgments

This work was partly financed by VINNOVA, the Swedish Agency for Innovation Systems.

References

[1] The DTN/SN project. Web page. Visited 2004-09-24.
URL: *http://www.sics.se/cna/dtnsn/*

[2] CST Group at FU Berlin. Scatterweb Embedded Sensor Board. Web page. Visited 2003-10-21.
URL: *http://www.scatterweb.com/*

[3] A. Dunkels. Full TCP/IP for 8-bit architectures. In *MOBISYS'03*, San Francisco, California, May 2003.

[4] A. Dunkels, J. Alonso, T. Voigt, and H. Ritter. Distributed TCP Caching for Wireless Sensor Networks. In *Proceedings of the Third Mediterranean Workshop on Ad Hoc Networks(Med-Hoc Net 2004)*, Bodrum, Turkey, June 2004.

[5] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *First IEEE Workshop on Embedded Networked Sensors*, 2004.

[6] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN'04), work-in-progress session*, Berlin, Germany, January 2004.

[7] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller. Connecting Wireless Sensornets with TCP/IP Networks. In *WWIC2004*, February 2004.

[8] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[9] L. M. Feeney, B. Ahlgren, A. Westerlund, and A. Dunkels. Spontnet: Experiences in configuring and securing small ad hoc networks. In *Proceedings of The Fifth International Workshop on Network Appliances(IWNA5)*, Liverpool, UK, October 2002.

[10] B. Kantor and P. Lapsley. Network News Transfer Protocol. RFC 977, Internet Engineering Task Force, February 1986.

[11] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. NSDI'04*, March 2004.

[12] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *First ACM Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, Atlanta, GA, USA, September 2002.

[13] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address allocation for private internets. RFC 1918, Internet Engineering Task Force, February 1996.