A Proposal for a Common Sensor Network Protocol

M. G. Gouda[1], A. Arora[2], J. A. Cobb[3], T. Herman[4], and S. Kulkarni[5]

[1] The University of Texas at Austin, gouda@cs.utexas.edu
[2] The Ohio State University, anish@cse.ohio-state.edu
[3] The University of Texas at Dallas, cobb@utdallas.edu
[4] The University of Iowa, herman@cs.uiowa.edu
[5] Michigan State University, sandeep@cse.msu.edu

Department of Computer Sciences,
The University of Texas at Austin,
Technical Report TR 04-54, October 2004

Abstract. We discuss the need for a common sensor network protocol, and present a preliminary design of such a protocol.

Key words: Protocol stack, sensor networks, hourglass architecture, message header, Internet Protocol

1 An Hourglass Protocol Architecture

In this section, we present an hourglass protocol architecture for sensor networks. We adopt this architecture in this proposal. It is important to note that this architecture is still in its early stages of development and so it is subject to further reviews and tuning as our work proceeds forward.

Before we present our protocol architecture, we first need to explain the structure of a sensor network that we assume as we develop this architecture. We assume that a sensor network consists of one gateway and at most 255 sensors. The gateway of a sensor network acts both as a central controler of the (at most 255) sensors in its network and as a gateway that connects its sensor network with other sensor networks or the rest of the Internet. Notice that a gateway can be the gateway for more than one sensor network. In this case, sensors that belong to different networks can not communicate directly. Rather they can communicate only via their common gateway.

Each sensor is a small computer that has a sensing board and an antenna, and it can communicate with other sensors in its network or with its gateway in a wireless fashion by broadcasting messages over radio frequency. Each sensor in a network has a unique identifier in the range 1..255. Identifier 0 in a sensor network is reserved for the gateway of that network.

The maximum number, 255, of sensors in a network needs some explanation. First, many applications (such as sensing in the home) requires a small number of sensors. For each such application, it is enough to install one network. Second, for an application that needs a large number of sensors, two or more networks, that share the same gateway or have different gateways, can be installed. Third, as discussed below, the different sensors in a network need to stay in close synchronization with one another, for example the clocks of these sensors need to stay reasonably close. This can be accomplished easily if the number of sensors in the network is relatively small.

The sensors in a network are not required to be identical. In fact, they may have different capacities and may be manufactured by different manufacturers. This flexibility necessitates that all sensors, regardless of their capacities or their manufacturers, are provided with a common protocol that allows the sensors to communicate with one another. We refer to this common protocol as the Sensor network Protocol or SP for short. To be exact, a sensor may have many protocols that perform different functions (such as message routing, clock synchronization, security, and

different sensing applications), but all these protocols reside on top of SP. Thus, the protocol stack in a sensor has the shape of an hourglass whose bottle-neck is SP.

It is important to note that SP is used only to support the communications between different sensors or between the sensors and the gateway in the same sensor network. The communications between different gateways (that belong to different sensor networks) or between a gateway and Internet clients and servers is supported by the Internet Protocol IP.

In several ways, the thought of SP evokes IP. Nevertheless, SP performs three functions that are different from those performed by IP. First, SP supports special routing patterns that are more appropriate for a sensor network, whereas IP supports general unicast and multicast routing patterns that are more appropriate for the Internet. Second, SP allows sensors to publicize and harmoniously reset their internal states so that any loss of synchronization between sensors in the same network can be detected and corrected as early as possible. This function is not supported by IP. Third, SP provides security, against message insertion and replay, in almost every message. This function is not provided by IP. We are now ready to present our current design of SP in more detail. (As mentioned above, this is an early design that needs to go through several revisions and tuning stages, before it evolves into a final design.)

2 The SP Header

Like IP, SP defines a header that needs to be attached to every message before this message is sent within a sensor network. The SP header of a message m consists of 10 bytes divided over nine fields. The fields of the SP header of a message m are specified as follows.

1. ID of a Sensor Network (1 Byte): This field specifies the least ordered byte in the ID of the sensor network in which message m is originated. We expect that the full ID of a sensor network consists of 4-5 bytes and that all sensors in a network and the gateway of the network know the ID of the network. Only the least ordered byte of the network ID is used to specify the sensor network in message m in order to keep the SP header short. (In computing the digest of message m in field 8 below, the full ID of the network is used instead of field 1.)

2. Routing Mode (2 Bits): This field defines the routing mode that needs to be used in routing message m. To fill this field, a choice is to be made from among three possible routing modes:

(a) Unicast: Message m is originated at some sensor in the network and it needs to be routed over the current routing tree towards the gateway of the network.
(b) Broadcast: Message m is originated at the gateway of the network and it needs to be routed over the current routing tree towards every sensor in the network.
(c) Flooding: Message m is originated at some sensor or at the gateway of the network and it needs to be routed to every sensor or the gateway that is within a specified time-to-live hops from the originator of m. The value of the time-to-live is specified by the originator of m in the (next) Routing Information field of the SP header.

Note that the Routing Mode field has two bits and so a fourth routing mode can be added to these three routing modes. For the time being, we leave this fourth routing mode to be identified in the future.

3. Routing Information (1 Byte): This field contains some information needed to route message m based on the previous Routing Mode field in the SP header. There are three cases to consider:
   (a) Routing Mode of m is "unicast": In this case, the Routing Information is the parent, in the current routing tree, of the sensor that just sent m.
   (b) Routing Mode of m is "broadcast": In this case, the Routing Information is the sensor that just sent m.
   (c) Routing Mode of m is "flooding": In this case, the Routing Information is the remaining time-to-live of message m.

4. Originator Identifier (1 Byte): This field stores the identifier of the sensor or gateway that originated message m.

5. Originator Distance (6 Bits): This field contains the number of hops that message m has made so far. When m is sent by its originator, the Originator Distance field in the SP header of m is 1. Each sensor that later forwards message m increments by one the value of its Originator Distance field before it forwards m. This field in the SP header of message m can be used by any sensor that receives m in two ways. First, the receiving sensor can use this field in comparing the (next) Originator Time field in the SP header of m with its own clock to decide whether m is a fresh (and not an old or replay) message. Second, the receiving sensor can use this field to keep track of

its shortest distance, in number of hops, from every other sensor in the network. This information can be used in routing future messages.

6. Originator Time (2 Bytes): This field stores the clock value of the originator of message m at the instant when m is originated. This field, along with the previous Originator Distance field, are used by any sensor that receives m to decide whether m is a fresh message (and so needs to be processed further) or an old or replay message (and so needs to be discarded).

7. Originator State (1 Byte): This field has a value in the range 0..255 that identifies the local state of the originator of message m at the instant when m is originated. This field is used in resetting the local states and times in a sensor network as follows. Periodically, the gateway of the sensor network broadcasts a message m, whose Originator State field is 0 indicating that every sensor in the network should reset its local state. When any sensor in the network receives this "reset" message m, the sensor resets its local state and assigns its local clock the value 0. This field can also have other uses in maintaining state and time synchronization between neighboring sensors in a sensor network.

8. Message Digest (2 Bytes): This field in the SP header of a message m contains the value of a message digest function applied to the concatenation of the following items: the "constant fields" in the SP header of message m, the rest of message m, the secret key that is shared by every sensor and the gateway in the network. This field is used by any final destination that receives message m to decide whether m was originated by a legitimate sensor or the gateway in the network (and not inserted into the network by a foreign sensor that does not belong to the network).

9. Next Protocol (1 Byte): This field defines the next protocol, after SP, that message m needs to be forwarded to, when m reaches (any of) its final destinations.

3 Future Work

Our work on reviewing, revising, and tuning SP continues. We divide our future work into the following six tasks.

- *Task 1.* Designing a routing protocol that supports SP: In this task, we design a protocol for maintaining a shortest-path routing tree from every sensor to the gateway in the sensor network. The maintained tree is used in routing unicast and broadcast messages. The protocol for maintaining the routing tree is periodically initiated by the gateway in the network. It uses flooding messages so that each sensor in the tree can compute its parent(s) in the new tree, in the case where the new tree happens to be different from the current tree.

- *Task 2.* Designing a security protocol that supports SP: Each sensor in the network has two security keys: one key that this sensor shares with all other sensor and the gateway and another key that this sensor shares only with the gateway. These keys need to be changed periodically. In this task, we design a protocol for periodically changing the security keys in each sensor and the gateway.

- *Task 3.* Designing a synchronization maintenance protocol that supports SP: In this task, we design a protocol for performing two functions: maintaining close synchronization between the local clocks and local states of neighboring sensors in the network and periodically resetting the local clocks and local states of all sensors in the network.

- *Task 4.* Implementing and evaluating the performance of SP: In this task, we implement SP and all its supporting protocols, that are designed in Tasks 1 through 3 above. We also experiment with these implementations and evaluate their performance in the sensor lab at the University of Texas at Austin and the testbed at the Ohio- State university.

- *Task 5.* Designing a translation between SP and IP: In this task, we design a translation from SP to IP and vice versa. This translation can be used by any gateway of a sensor network to transform one or several SP messages that are generated in its network into a single IP message that the gateway needs to forward over the Internet (either to the gateway of another sensor network or to some Internet client or server). This translation can also be used by the gateway of a sensor network to transform an IP message that the gateway receives over the Internet (either from the gateway of another sensor network or from some Internet client or server) into one or several SP messages

that the gateway needs to forward to the sensors in its network.

– *Task 6.* Designing sensing transport protocols over SP: The routing, security, and synchronization protocols designed in Tasks 1 through 3 above constitute the smallest set of protocols that are needed to support SP. We expect, nevertheless, that other (in particular transport) protocols will be needed to support the different application protocols in sensor networks. Some of these transport protocols, for instance, can be invoked to reserve needed resources or guarantee quality of service for the application protocols. We are planning on designing these transport protocols in this task.

4 Related Work

Our effort to adopt an hourglass architecture for sensor network protocols was inspired by the hourglass architecture for the Internet protocols as discussed in [4] and [3]. Our current design of SP was inspired to a large extent by the well-documented design of IP version 4 in [13] and IP version 6 in [5]. Our plan to design a routing protocol that supports SP is greatly influenced by the routing protocols in [15], [9], [16], [1], [8] and [2] that were proposed earlier for ad-hoc networks. Our proposal to design a security protocol that supports SP is motivated by the earlier security protocol proposals in [11], [10], [17], [12]and [7]. Finally, our decision to design a synchronization protocol that supports SP was inspired by the two time synchronization protocols proposed in [6] and [14].

5 Acknowledgment

These pages (more or less) were part of a proposal that we submitted to the NSF in May 2004. Unfortunately, the proposal was not funded. We wish to thank Professors David Culler and Scott Shenker for encouraging us to develop this document in order to expose our ideas concerning SP to the public.

References

1. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (ACM MobiCom)*, pages 85–96, Rome, Italy, July 2001.

2. Y. Choi, M. G. Gouda, H. Zhang, and A. Arora. Routing on a Logical Grid in Sensor Networks. Technical Report TR04-49, Department of Computer Sciences, The University of Texas at Austin, 2004.
3. D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *SIGCOMM*, pages 106–114, Stanford, CA, Aug. 1988. ACM.
4. S. Deering. Watching the waist of the protocol hourglass. *Keynote address at ICNP '98*, October 1998.
5. S. Deering and R. Hinden. Internet Protocol Version 6 (IPv6) Specification. *RFC 2460*, December 1998.
6. J. Elson and K. Romer. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I)*, Princeton, New Jersey, October 2002.
7. M. Gouda, Y. Choi, and A. Arora. Antireplay protocols for sensor networks. *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, (ed. Jie Wu), CRC*, 2005.
8. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocols for Wireless Microsensor Networks. In *Proceedings of Hawaiian Int'l Conf. on Systems Science*, January 2000.
9. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000)*, Boston, Massachusetts, August 2000.
10. S. Kent and R. Atkinson. IP Authentication Header. *RFC 2402*, November 1998.
11. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *RFC 2401*, November 1998.
12. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of the 7th annual international conference on Mobile Computing (MobiCom 2001)*, pages 189–199, New York, 2001.
13. J. Postel. Internet protocol. *RFC 791*, September 1981.
14. K. Romer. Time Synchronization in Ad Hoc Networks. In *Proceedings of MobiHoc 2001, ACM*, October 2001.
15. A. Woo, T. Tony, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of ACM SenSys*, Los Angelos, CA, 2003.
16. Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad-hoc routing. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (ACM MobiCom)*, Rome, Italy, July 2001.
17. L. Zhou and Z. Haas. Securing Ad Hoc Networks. *IEEE Networks Special Issue on Network Security*, November/December, 1999.