Lynch, N. A. and Livadas, C. A Reliable Broadcast Scheme for Sensor Networks. Technical Report MIT/CSAIL/TR-???, Computer Science and Artificial Intelligence Lab., MIT, Cambridge, MA, Aug. 2003.

A Reliable Broadcast Scheme for Sensor Networks

Carolos Livadas Nancy A. Lynch
Lab. for Computer Science
Massachusetts Institute of Technology

August 11, 2003

Abstract

In this short technical report, we present a simple yet effective reliable broadcast protocol for sensor networks. This protocol disseminates packets throughout the sensor network by flooding and recovers from losses resulting from collisions by having hosts retransmit packets whenever they notice that their neighbors have fallen behind. Such retransmissions serve to flood the appropriate packets throughout the regions of the sensor network that did not receive the given packets as a result of prior flooding attempts.

1 Overview

In this short technical report, we present a simple reliable broadcast scheme for sensor networks. We presume that the sensor network consists of a set of hosts that are scattered in space and are stationary. Moreover, we presume that the hosts are capable of communicating wirelessly with their neighbors (within a certain radius) and that the hosts and their wireless connections form a connected graph. For simplicity, we presume that there is only one transmission source s and that our broadcasting scheme is used to broadcast a sequence of packets from s to all other hosts.

Our reliable broadcasting scheme consists of two mechanisms: packets are disseminated throughout the sensor network by flooding and looses are recovered by having hosts retransmit packets whenever they notice that their neighbors have fallen behind. The first mechanism floods a packet p to all the hosts reachable from s by having each host retransmit p whenever it first receives it. Since the wireless connectivity of all hosts forms a connected graph, when no losses occur p is flooded to all hosts in the sensor network.

The second mechanism is responsible for recovering from losses resulting from the collision (overlap) of the wireless transmissions of neighboring hosts. Each host h periodically transmits its *frontier* sequence number; that is, the sequence number of the most recent packet p' sent by s such that h has received all packets up to and including p'. Upon receiving this information, if a neighbor h' of h has a larger frontier sequence number than the frontier sequence number advertised by h, then h' transmits all the packets that it has received and whose sequence numbers are greater than the frontier sequence number advertised by h. Thus, the given packets are forwarded to h and, subsequently, flooded to the region of the graph of hosts that have, potentially, not received the given packets.

Figure 1 Definition of Packet Operations

$P =$ Set of packets such that for all $p \in P$:
  $type(p) \in \{\texttt{DATA}, \texttt{UPDT}\}$
  if $type(p) = \texttt{DATA}$ **then**
    $source(p) \in H$
    $seqno(p) \in \mathbb{N}$
    $data(p) \in \{0,1\}^*$
  if $type(p) = \texttt{UPDT}$ **then**
    $source(p) \in H$
    $frontier(p) \in \mathbb{N}$

Figure 2 The HOST_h Automaton — Signature

Parameters:

$h \in H, \texttt{SOURCE} \in H, \texttt{UPDT-PERIOD} \in \mathbb{R}^+$

Actions:

input
  $\texttt{brecv}_h(p)$, for $p \in P$
output
  $\texttt{bsend}_h(p)$, for $p \in P$
  $\texttt{updt}_h$
internal
  $\texttt{bcast}_h$
time-passage
  $\nu(t)$, for $t \in \mathbb{R}^{\geq 0}$

2 Formal Model

We abstractly model the physical system involving a set of hosts that communicate wirelessly with their neighbors as the interaction of a set of host automata (one for each host) and a single environment automaton that encapsulates the behavior of the hosts' environment. We let H denote the set of all hosts in the sensor network and P denote the set of all packets that could ever be broadcast by any of the hosts. Figure 1 presents the operations supported by the packets in P.

We model each host $h \in H$ by the automaton HOST_h, the environment by the automaton ENV, and the complete sensor network as the composition $\text{ENV} \times \prod_{h \in H} \text{HOST}_h$.

2.1 The HOST_h Automaton

Figures 2 and 3 specify the signature, the variables, and the discrete transitions of the HOST_h automaton. The parameter $h \in H$ identifies the host whose behavior the automaton HOST_h models. In order to provide the appropriate context, the description of the parameters `SOURCE` and `UPDT-PERIOD` is deferred to appropriate places within the description of its variables and actions.

The variable $now \in \mathbb{R}^{\geq 0}$ denotes the time that has elapsed since the beginning of an execution of HOST_h. The variable *received* is the set of packets that the host h has received. The variable *bqueue* is the set of packets whose transmission is pending at the host h. The variable *updt-time* is the time at which the next update packet of host h is scheduled for transmission. The variable *seqno* is the sequence number of the next packet to be broadcast by the host h. The derived variable *frontier* is the frontier sequence number of the host h.

The input action $\texttt{brecv}_h(p)$ models the reception of a packet p by the host h. If p is a `DATA` packet that the host h has not yet received, then the $\texttt{brecv}_h(p)$ action archives p and adds it to the broadcast queue *bqueue* so that it may subsequently be forwarded to the neighbors of h. Notice that p is forwarded by h only if h has not previously received it. If p is an `UPDT` packet and the frontier sequence number of h is greater than the frontier sequence number advertised by p, then $\texttt{brecv}_h(p)$ adds to the broadcast queue *bqueue* all the packets that it has received whose sequence number is greater than the frontier sequence number advertised by p.

The internal action $\texttt{updt}_h$ models the scheduling of the transmission of an update packet that announces h's current frontier sequence number to its neighbors. The $\texttt{updt}_h$ action is enabled

Figure 3 The $\textsc{host}_h$ Automaton — Variables and Discrete Transitions

Variables:

$now \in \mathbb{R}^{\geq 0}$, initially $now = 0$
$received \subseteq P$, initially $received = \emptyset$
$bqueue \subseteq P$, initially $bqueue = \emptyset$
$updt\text{-}time \in \mathbb{R}^{\geq 0}$, initially $updt\text{-}time \in [0, \texttt{UPDT-PERIOD}]$
$seqno \in \mathbb{N}$, initially $seqno = 0$

Derived Variables:

$frontier = \max(\{i \in \mathbb{N} \mid \exists p \in received, seqno(p) = i \wedge \forall i' \mathbb{N}, i' < i, \exists p' \in received, seqno(p') = i'\})$

Discrete Transitions:

output $\texttt{bsend}_h(p)$
pre $p \in bqueue$
eff $bqueue \setminus= \{p\}$

internal $\texttt{updt}_h$
pre $now = updt\text{-}time$
eff $bqueue \cup= \{comp\text{-}updt\text{-}pkt()\}$
$updt\text{-}time := now + \texttt{UPDT-PERIOD}$

internal $\texttt{bcast}_h$
pre $h = \texttt{SOURCE}$
eff $bqueue \cup= \{comp\text{-}data\text{-}pkt()\}$
$seqno := seqno + 1$

input $\texttt{brecv}_h(p)$
where $type(p) = \texttt{DATA}$
eff **if** $p \notin received$ **then**
$bqueue \cup= \{p\}$
$received \cup= \{p\}$

input $\texttt{brecv}_h(p)$
where $type(p) = \texttt{UPDT}$
eff **if** $frontier(p) < frontier$ **then**
foreach $p' \in received : frontier < seqno(p')$ **do:**
$bqueue \cup= \{p'\}$

time-passage $\nu(t)$
pre $now + t \leq updt\text{-}time$
eff $now := now + t$

when the current time *now* is equal to the update packet transmission deadline *updt-time*. The $\texttt{updt}_h$ action composes the update packet from the current state of the automaton $\textsc{host}_h$, adds it to the broadcast queue *bqueue*, and resets the update packet transmission deadline *updt-time* to `UPDT-PERIOD` time units in the future. The operation *comp-updt-pkt*() returns an update packet that is consistent with the current state of $\textsc{host}_h$.

The internal action $\texttt{bcast}_h$ models the generation of a new packet to be transmitted by the source of the transmission. The $\texttt{bcast}_h$ action is enabled when the host h is the transmission source `SOURCE`. Its effects are to compose a data packet, to add it to the broadcast queue *bqueue*, and to increment the sequence number *seqno* of the next packet to be transmitted. The operation *comp-data-pkt*() returns the next data packet in the sequence of data packets to be transmitted by h; that is, a data packet whose sequence number is equal to *seqno*.

The output action $\texttt{bsend}_h(p)$ models the transmission of the packet p by the host h. This action is enabled when the packet p is in the broadcast queue *bqueue*. Its effects are to remove p from the broadcast queue *bqueue*.

The time-passage action $\nu(t)$ models the passage of t time units. The $\nu(t)$ action is enabled when h's deadline for transmitting an update packet does not precede the point in time $now + t$. Its effects are to increment the variable *now* by t time units.

2.2 The $\textsc{host}_h$ Automaton

Figures 4 and 5 specify the signature, the variables, and the discrete transitions of the $\textsc{env}$ automaton. This automaton models the environment of the sensor network. The $\textsc{env}$ automaton also maintains the sensor network topology. This topology is presumed to be fixed.

The variable $now \in \mathbb{R}^{\geq 0}$ denotes the time that has elapsed since the beginning of an execution of $\textsc{env}$. The variables $dqueue(h)$, for $h \in H$, are the delivery queues for each host h; that is, for any packet $p \in dqueue(h)$, it is the case that the delivery of p to h is pending. Each set $neighbors(h)$, for $h \in H$, indicates the neighboring hosts of h in the sensor network; that is, the set of hosts that are reachable through h's wireless transmissions. The sets $neighbors(h)$, for $h \in H$, dictate the

Figure 4 The ENV Automaton — Signature

Actions:	
input	**time-passage**
$\mathtt{bsend}_h(p)$, for $p \in P$	$\nu(t)$, for $t \in \mathbb{R}^{\geq 0}$
output	
$\mathtt{brecv}_h(p)$, for $p \in P$	
$\mathtt{bdrop}_h(p)$, for $p \in P$	

Figure 5 The ENV Automaton — Variables and Discrete Transitions

Variables:

$now \in \mathbb{R}^{\geq 0}$, initially $now = 0$
$dqueue(h) \subseteq P$, for $h \in H$, initially $dqueue = \emptyset$, for all $h \in H$
$neighbors(h) \subseteq H$, for $h \in H$, initially $neighbors(h) \subseteq H$, for all $h \in H$,
such that H and $neighbors(h)$, for $h \in H$, form a connected graph

Discrete Transitions:

input $\mathtt{bsend}_h(p)$	**output** $\mathtt{bdrop}_h(p)$
eff **foreach** $h' \in neighbors(h)$ **do:** $dqueue(h') \cup= \{p\}$	**pre** $p \in dqueue(h)$ **eff** $dqueue(h) \setminus= \{p\}$
output $\mathtt{bdlvr}_h(p)$	**time-passage** $\nu(t)$
pre $p \in dqueue(h)$ **eff** $dqueue(h) \setminus= \{p\}$	**pre** None. **eff** $now := now + t$

topology of the sensor network.

The input action $\mathtt{bsend}_h(p)$, which models the wireless transmission of the packet p by h, adds the packet p to the delivery queues of all the neighbors of h. Thus, we model the omni-directional wireless transmission of p by h as a parallel point-to-point transmission from h to all of its neighbors.

The output action $\mathtt{bdlvr}_h(p)$ models the delivery of the packet p to h. The $\mathtt{bdlvr}_h(p)$ is enabled when p is in the delivery queue $dqueue(h)$ of h. Its effects are to remove p from the delivery queue $dqueue(h)$ of h.

The output action $\mathtt{bdrop}_h(p)$ models the loss of the packet p while being delivered to h. By modeling the omni-directional wireless transmission of packets as multiple point-to-point transmissions, we consider the transmission failure to any single neighbor as a distinct loss. We presume that losses are due solely to collisions, *i.e.*, when neighboring hosts attempt to transmit concurrently. More complicated models of the system and its behavior may model power failures, crashes, and, potentially, sensor/host mobility.

The time-passage action $\nu(t)$ models the passage of t time units. The $\nu(t)$ action is enabled at any point in time and its effects are to increment the variable now by t time units.

3 Informal Performance Analysis

In this section, we informally analyze the performance of our reliable broadcast scheme for sensor networks. Let D denote the diameter of the sensor network in terms of hop-counts; that is, any sensor is at most D hops away from any other sensor in the network. Moreover, let $\overline{d}$ be an upper bound on the amount of time it takes for a host to wirelessly transmit a packet. Finally, let $f \in \mathbb{N}$ be a bound on the total number of losses suffered during any particular execution of the system.

Now, consider the effect that a packet drop has on the dissemination of a packet throughout the sensor network. Let h and h' be any two neighbors in the sensor network and suppose that h' is reachable from the transmission source s only through h'. Now, consider the scenario in which a data packet p is dropped while being transmitted from h to h'. Since h may only transmit a packet a second time as a response to an update packet, h' may only recover the packet after successfully transmitting an update packet that instigates the retransmission of p by h.

First, consider the case where this update packet indeed instigates the retransmission of p by h. In such a case, the additional delay in forwarding p that is incurred by the loss of the original transmission of p from h to h' is equal to $\texttt{UPDT-PERIOD}+\overline{d}$; this corresponds to the latest point in time that h may receive the next update packet of h'. Once this update packet is received, h retransmits p and, presuming no additional packets are dropped, p is subsequently flooded throughout the region of the sensor network reachable through h'. The dissemination of p from h to h' may further be delayed either if the update packet from h' to h or the retransmission of p from h to h' is dropped. Each such drop would delay the dissemination of p from h to h' by UPDT-PERIOD additional time units. It follows that each packet drop may delay a packet's recovery by at most $\texttt{UPDT-PERIOD} + \overline{d}$ time units.

In our example of the packet p being dropped while being transmitted from h to h', it is possible for an update packet from h' not to instigate the retransmission of p by h. This may occur when both h and h' share the loss of a packet p' that was transmitted by s earlier than p. In such a case, the recovery of p may rely on the recovery of p', which may itself be delayed due to packet losses. Once p' is recovered however, the recovery of p proceeds as described above. In fact, a cascade of packet recoveries may need to be carried out before p may be recovered. Once again, however, each such recovery is delayed by at most $\texttt{UPDT-PERIOD} + \overline{d}$ per loss.

Presuming that the sensor network suffers at most $f \in \mathbb{N}$ packet losses, the worst-case delay incurred during the recovery of p would be $f(\texttt{UPDT-PERIOD} + \overline{d})$. Thus, the worst-case delay in broadcasting p would be $D\overline{d}+f(\texttt{UPDT-PERIOD}+\overline{d})$, where $D\overline{d}$ is the worst-case propagation delay of p throughout the sensor network and $f(\texttt{UPDT-PERIOD} + \overline{d})$ is the delay incurred due to the f possible losses.

4 Discussion and Conclusions

The reliable broadcast scheme presented in this report is a very simple yet effective scheme that guarantees the delivery of packets from a single source to the other hosts in a sensor network. Our scheme is particularly effective for low bandwidth sensor network transmissions, which is the type of application for which it was designed.

Although simple and effective, our reliable broadcast scheme has some limitations. The hosts are required to archive all the packets they receive. This may not be feasible in an actual sensor network. Moreover, the straightforward extension to the case where there are multiple sources the size of the update packets is proportional to the number of sources. This may make the transmission of update packets prohibitively expensive and fault-prone. Finally, the protocol may not be well suited to high bandwidth transmissions. The combination of high bandwidth transmissions and the fact that wireless communication is highly lossy may induce frequent frontier sequence number discrepancies and result in frequent retransmission bursts. These are limitations that are worth thinking about and resolving.