Routing on a Logical Grid in Sensor Networks

Young-ri Choi[1], Mohamed G. Gouda[1], Hongwei Zhang[2] and Anish Arora[2]
[1] The University of Texas at Austin, Austin, TX 78712-0233, U.S.A.
{yrchoi, gouda}@cs.utexas.edu
[2] The Ohio State University, Columbus, OH 43210-1277, U.S.A.
{zhangho, anish}@cse.ohio-state.edu

UTCS Technical Report TR-04-49

Abstract

We present a protocol for routing data messages from any sensor to the base station in a sensor network. The protocol maintains an incoming spanning tree whose root is the base station. The spanning tree is constructed as follows. First, each sensor in the network is assigned a unique identifier as if the sensors form a logical two-dimensional grid. Second, each sensor, other than the base station, uses its own identifier to compute the identifiers of its "potential parents" in the spanning tree. Third, the base station starts to periodically send "connected messages". When a sensor receives a connected message from anyone of its potential parents, the sensor makes this potential parent its parent in the tree and starts to periodically send connected messages. This routing protocol has three advantages over earlier protocols: overhead of the protocol is very small, the protocol balances the load over the whole network, and the protocol has nice fault-tolerance and security properties. We have implemented this protocol over a sensor network that consisted of about 50 MICA2 motes and observed that the protocol delivers 72% – 99% of the messages to the base station under heavy and bursty traffic (that generated 100 data messages per 15 seconds). We also ran the same experiment on a version of the distance vector routing protocol and observed that this protocol can delivery only 34% of the messages to the base station.

I. Introduction

A sensor is a battery-operated small computer with an antenna and a sensing board that can sense magnetism, sound, heat, etc. Sensors in a network can use their antennas to communicate in a wireless fashion by broadcasting messages over radio frequency to neighboring sensors in the same network. Due to the limited range of radio transmission, sensor networks are usually

multi-hop. Sensor networks can be used for military, environmental or commercial applications such as intrusion detection [1], coutersniper system [2], disaster monitoring [3] and habitat monitoring [4].

A sensor network usually supports two communication patterns between the sensors in the network: unicast and broadcast. In the unicast pattern, any sensor in the network can send a message whose ultimate destination is the base station. This pattern is used, for example, when a sensor senses an event and needs to report the event to the base station. In the broadcast pattern, the base station can send a message whose ultimate destination is every sensor in the network. This pattern is used, for example, to tune the parameters in the different sensors or reset the whole network.

It is difficult to design routing protocols for sensor networks. This is because these routing protocols need to overcome the special challenges that are posed by sensor networks. First, a sensor network has limited resources. Examples of these resources are the memory available in each sensor, the energy remaining for each sensor, and the communication capacity of the sensor network. Any routing protocol for sensor networks should not consume a large fraction of the resources of the sensor networks. For example, sensors should not need to store a large routing table. Also, sensors should not need to send large routing messages frequently.

Second, sensors in a sensor network can be unreliable. For example, some sensors in the network may fail-stop as they run out of their energy or they are physically damaged. Any routing protocol for sensor networks should be able to recover from the situation where a sizable fraction of the sensors fail-stop. Moreover, the recovery should be fast, taking several seconds (rather than minutes).

In this paper, we present a routing protocol, called the logical grid routing protocol, that can overcome the challenges of sensor networks. First, the protocol is simple and so it consumes a small percentage of the resources of its network. In particular, the protocol requires that every sensor in the network sends only one routing message every 20 seconds and stores no more than 10 – 15 bytes of routing information. Also, each routing message has no more than 20 – 30 bytes. Second, we show (by simulation) that even if 50% of the sensors in a network fail-stop,

84% of the remaining sensors can still route data messages.

Our protocol maintains an incoming spanning tree whose root is the base station. The spanning tree is constructed as follows. First, each sensor in the network is assigned a unique identifier as if the sensors form a logical two-dimensional grid. Second, each sensor, other than the base station, uses its own identifier to compute the identifiers of its "potential parents" in the spanning tree. Third, the base station starts to periodically send "connected messages". When a sensor receives a connected message from anyone of its potential parents, the sensor makes this potential parent its parent in the tree and starts to periodically send connected messages.

A sensor in the network cannot have a parent in the tree if all its potential parents fail-stop. To solve this problem, we extend the logical grid routing protocol such that a sensor can have a "foster parent" in the tree when all its potential parents fail-stop and the sensor receives a connected message from any other sensor in the network.

Connected messages that are periodically sent by a sensor are very small: each message consists of 2 – 4 bytes. Therefore, the broadcast messages from the base station can be piggybacked on the connected messages. We extend the logical grid routing protocol sightly to piggyack a broadcast message on the connected messages.

Clearly, the logical grid routing protocol is a *localized* algorithm as characterized in [5], [6] and [7]. Each sensor in the network computes its parent in the tree based solely on its sensor identifier in the logical grid. The spanning tree whose root is the base station can be maintained in the network as each sensor maintains its parent locally. This localized algorithm provides simplicity and scalability for sensor networks.

The rest of the paper is organized as follows. In Section II, we discuss a logical grid and present an algorithm to compute potential parents for each sensor. We then present the logical grid routing protocol in Section III and specify how sensors in the network route data messages in Section IV. In Section V, we discuss foster parents. In Section VI, we present a broadcast protocol. We then show the experimental results of the logical grid routing protocol in Section VII, and discuss the advantages and limitations in Section VIII. We discuss the related work in Section IX and finally make concluding remarks in Section X.

II. Logical Grid and Potential Parents

We consider a sensor network where sensors are deployed at arbitrary physical locations, but they are named as if they form an $M * N$ logical grid. In this network, each sensor is identified by a pair (i,j), called the sensor identifier in the logical grid, where i= 0..$M-1$ and j= 0..$N-1$. The base station in this network is sensor (0,0).

(It is advantageous to select the base station to be the grid point ($M/2$, $N/2$) at the middle of the logical grid. In this case, the maximum number of hops to be traveled by data messages from any sensor to the base station is minimized. Moreover, the probability that the base station can be separated from the rest of the network is also minimized. Nevertheless, selecting the base station to be in the middle of the logical grid will complicate the algorithm, discussed below, for computing potential parents for each sensor. Thus, to keep our presentation simple, we choose the base station to be the grid point (0,0).)

When a sensor (i,j) has a data item and wishes to send this data item to its final destination, the base station (0,0), sensor (i,j) forwards the data item to a far away sensor (i',j') that satisfies one of the following three conditions:

i) $i > i'$ and $j > j'$

ii) $i = i'$ and $j > j'$

iii) $i > i'$ and $j = j'$

The transmission of the data item from sensor (i,j) to sensor (i',j') is called *one hop*.

The reason that sensor (i,j) needs to forward the date item to a far away sensor (i',j') is to reduce the total number of hops that the data item needs to travel from its original source (any sensor in the network) to the ultimate destination (the base station of the network).

The requirement that sensor (i,j) forwards its data item to a "far away" sensor (i',j') can be stated formally as follows:

$$(i - i') + (j - j') = H$$

where H is a small positive integer, called the *hop size*. In this case, sensor (i',j') is called a *potential parent* of sensor (i,j) in the incoming routing tree whose root is the base station (0,0).

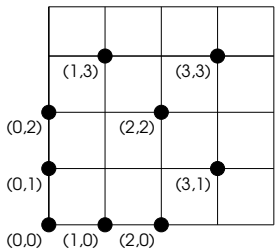


Fig. 1. A 5*5 logical grid where the hop size is 2

In general, sensor (i,j) can have up to $H + 1$ potential parents; These potential parents are

sensor (i, $j - H$)

sensor ($i - 1$, $j - H + 1$)

...

sensor ($i - H + 1$, $j - 1$)

sensor ($i - H$, j)

It follows from the above discussion that if H is chosen to be two, then each sensor (i,j) in the logical grid has up to three potential parents. For example, referring to a logical grid in Fig. 1, the potential parents of sensor (3,3) are sensors (1,3), (2,2), (3,1). However, not every sensor has three potential parents in this case. For example, the base station (0,0) has no potential parent. Each of the two sensors (0,1) and (1,0) has only one potential parent, namely the base station (0,0). Also, sensor (0,2) has two potential parents (0,1), (0,0), and sensor (2,0) has two potential parents (0,0), (1,0).

The following algorithm can be used by any sensor (i,j) to compute set P of its potential parents in an $M * N$ logical grid where the hop size is H.

```
Algorithm

inputs    M,N : integer,          // M*N grid
          H   : integer,          // hop size
          i   : 0..M-1,
          j   : 0..N-1

outputs:  set P of potential parents of sensor (i,j)
          in an (M*N) grid whose hop size is H
```

```
variables u,v : 0..H

begin
      u := 0;
      do u <= H -> v := H-u;
                    if i-u>=0 and j-v>=0 -> add sensor (i-u, j-v) to set P
                    [] i-u<0  and j-v>=0 -> add sensor (0, j-v) to set P
                    [] i-u>=0 and j-v<0  -> add sensor (i-u, 0) to set P
                    [] i-u<0  and j-v<0  -> skip
                    fi;
                    u := u+1
      od;
      remove sensor (i,j) from set P
end
```

Let us summarize what we have done in this section. We started with a sensor network where the sensors are deployed at arbitrary physical locations. We then imposed an $M * N$ logical grid on this network. (This is accomplished by assigning a distinct pair (i,j) to each sensor in the network.) We then selected a hop size H and used this H to define the set of potential parents for each sensor in the network. In order to establish that the imposed logical grid and the selected hop size are *valid*, it is sufficient to check that each sensor can exchange data items with everyone of its potential parents. If there is some sensor that cannot exchange data items with everyone of its potential parents, then the imposed logical grid needs to be changed (by reassigning different identifiers to the sensors in the network) or the hop size needs to be reduced.

From now on, we assume that the imposed logical grid on the sensor network and the selected hop size are valid.

III. The Logical Grid Routing Protocol

The purpose of the logical grid routing protocol is to build and maintain an incoming spanning tree whose root is the base station (0,0). Each sensor (i,j) chooses one of its potential potential parents to be its parent in this tree. This tree is to be used to route data items from any sensor in the network to the base station (0,0).

Initially, only the base station (0,0) is in the spanning tree, and so the base station periodically sends a message of the form, connected(0,0), every T seconds.

When a sensor (i,j), that is currently not connected to the tree, receives a connected(i',j') message and checks that sensor (i',j') is one of its potential parents, sensor (i,j) becomes *connected* to the tree and makes sensor (i',j') its parent. From this point on, sensor (i,j) sends a connected(i,j) message periodically every T seconds.

When a sensor (i,j), that is currently connected to the tree and whose parent is sensor (i',j'), receives a connected(i'',j'') message and checks that sensor (i'',j'') is one of its potential parents, sensor (i,j) remains connected to the tree, but changes its parent in the tree from sensor (i',j') to sensor (i'',j''). In this case, sensor (i,j) continues to send a connected(i,j) message periodically every T seconds.

When a sensor (i,j), that is currently connected to the tree and whose parent is sensor (i',j'), does not receive any connected(i'',j'') message from any of its potential parents for a period $3 * T$ seconds, sensor (i,j) concludes that it is no longer connected to the tree and stops sending connected(i,j) messages. Later when sensor (i,j) receives a connected(i'',j'') message and checks that sensor (i'',j'') is one of its potential parents, sensor (i,j) becomes connected again to the tree and makes sensor (i'',j'') its parent in the tree.

In the above description of the protocol, we implied that the time period between two consecutive connected messages from the same sensor is T seconds. In order to reduce the likelihood that connected messages from adjacent sensors collide with each other, we make the time period between two consecutive connected messages from the same sensor random whose average is T seconds. Thus, when a sensor sends a connected message, the sensor computes a random period r after which the sensor needs to send the next connected message. Period r is computed by executing the statement

```
r := rand
```

This statement assigns r a random value taken from the range 1, 2, 3, ... , $2*T-1$.

A specification of the base station is as follows.

```
sensor (0,0)                // base station
const T   : integer         // avg interval to send next connected msg
```

```
var   r   : 1..2*T - 1       // random interval whose avg length is T

begin
   time-out expires -> send connected(0,0);
                       r := rand; time-out after r
end
```

The base station has only one action. When the timeout of sensor (0,0) expires, sensor (0,0) sends a connected(0,0) message and schedules to send the next connected(0,0) message after a random period r whose average length is T seconds. (Initially the timeout in sensor (0,0) is ready to expire.)

We mentioned earlier that when a sensor does not receive a connected message from any of its potential parents for the time period $3 * T$ seconds, the sensor recognizes that it is no longer connected to the spanning tree. This feature is implemented by providing each sensor (i,j), where $i \neq 0$ or $j \neq 0$, with a variable trc whose value is in the range 0..3. When sensor (i,j) receives a connected message from any of its potential parents, trc is assigned the value 3. Every time sensor (i,j) times-out to send a connected message, the value of variable trc is decremented by one using the assignment statement

```
trc := max(trc-1,0)
```

When the value of variable trc becomes zero, sensor (i,j) recognizes that it is no longer connected to the spanning tree.

A specification of sensor (i,j), where $i \neq 0$ or $j \neq 0$, is as follows.

```
sensor (i,j)                    // a sensor (i,j) where i=!0 or j!=0

const P   : set of potential parents of sensor (i,j),
      T   : integer             // avg interval to send next connected msg

var   pid : an element from P,  // parent identifier
      trc : 0..3,               // time to remain connected, initially 0
      r   : 1..2*T - 1,         // random interval whose avg length is T
      x   : 0..M-1,
      y   : 0..N-1

begin
   rcv connected(x,y) ->
      if (x,y) in P    -> pid := (x,y);          // choose new parent
                          if trc=0 -> r := rand; time-out after r
```

```
                          [] trc>0 -> skip
                          fi; trc := 3
      [] !((x,y) in P) -> skip
      fi

[] time-out expires ->    trc := max(trc-1,0);
                          if trc>0 -> send connected(i,j);
                                      r := rand; time-out after r
                          [] trc=0 -> skip       // lose parent
                          fi
end
```

Sensor (i,j) has two actions. The first action in sensor (i,j) is executed when sensor (i,j) receives a connected(x,y) message. In this case, there are two possibilities to consider. First, sensor (x,y) is one of the potential parents of sensor (i,j). In this case, sensor (i,j) makes sensor (x,y) its parent. Then if the time to remain connected to the tree, trc, is zero, sensor (i,j) schedules to send a connected(i,j) message after a random period r whose average length is T seconds. Otherwise, sensor (i,j) has already scheduled to send the next connected(i,j) message. Sensor (i,j) then sets trc to 3. Second, sensor (x,y) is not one of the potential parents of sensor (i,j). In this case, sensor (i,j) discards the connected(x,y) message.

The second action in sensor (i,j) is executed when the timeout of sensor (i,j) expires. In this case, sensor (i,j) decreases the time to remain connected to the tree, trc, by one. If the resulting trc is bigger than zero, sensor (i,j) sends a connected(i,j) message and schedules to send the next connected(i,j) message after a random period r whose average length is T seconds. On the other hand, if the resulting trc is zero, sensor (i,j) loses its parent and is no longer connected to the tree.

Notice that in the logical grid protocol, the parent of a sensor at any time is the last potential parent from which this sensor received a connected message. In other words, a sensor keeps changing its parent whenever it receives a connected message from another potential parent. This feature provides two nice properties: load balancing and fast fault recovery. First, load balancing is achieved, since the data messages, that are generated at the same sensor, are likely to follow different routes to the base station. Second, fast fault recovery is achieved when the current parent of a sensor fail-stops. In this case, the sensor replaces this parent as soon as it receives a

connected message from another potential parent. (Note that this feature may cause the arrival of data messages at the base station out of order. However, this is not a problem in sensor networks since most data messages are tagged with the realtime of when they were generated.)

IV. Routing Data Messages

To complete our specification of the logical grid routing protocol in Section III, we need to specify how this protocol is used to route data messages.

When a sensor (i,j), other than the base station, has a data message to route to the base station (this data message could have been generated by sensor (i,j) itself or it could have been forwarded to sensor (i,j) from another sensor that considers sensor (i,j) its parent in the spanning tree), sensor (i,j) first checks whether or not it is connected to the spanning tree. If the value of variable trc in sensor (i,j) is more than zero, sensor (i,j) concludes that it is connected to the tree and that the identifier of its current parent in the tree is in its variable pid. In this case, sensor (i,j) sends the message after attaching the value of variable pid to the message. On the other hand, if the value of variable trc in sensor (i,j) equals zero, then sensor (i,j) recognizes that it is not connected to the tree and drops the message.

The following two actions need to be added to the specification of sensor (i,j), where $i \neq 0$ or $j \neq 0$, in Section III.

```
   {generate data msg} ->
       if trc>0 -> send data(pid)                 // forward data msg to parent
       [] trc=0 -> skip                           // drop data msg
       fi

[] rcv data(x,y) ->
       if trc>0 and i=x and j=y -> send data(pid) // forward data msg to parent
       [] trc=0 or i!=x or j!=y -> skip           // drop msg
       fi
```

When the base station (0,0) receives any data(i,j) message, the base station accepts the data message even if (i,j) is not (0,0). This is because the ultimate destination of all data messages is the base station. Note that the same data message may be received by the base station more than once, since the data message is still forwarded along the spanning tree until it reaches the base station. Thus, the snooping feature of the base station can only increase the probability that

every data message is received (at least once) by the base station.

The following action needs to be added to the specification of the base station in Section III.

```
rcv data(x,y) ->  {accept data msg}
```

V. Foster Parents

According to the logical grid routing protocol (described in Section III), a sensor has a parent in the spanning tree as long as this sensor keeps on receiving connected messages from one or more of its potential parents. Thus, if at least one of the potential parents of a sensor is alive and connected to the spanning tree, the sensor remains connected to the tree. Unfortunately, it is possible that all the potential parents of a sensor fail-stop. When this happens, the sensor no longer receives any connected messages from any of its potential parents and so it becomes disconnected from the spanning tree.

To solve this problem, we need to extend the logical grid routing protocol such that a sensor remains connected to the spanning tree even if all its potential parents has fail-stopped. We describe this extension next.

When a sensor (i,j) has no parent and receives a connected message from some sensor (i',j'), which is not a potential parent of sensor (i,j), sensor (i,j) makes sensor (i',j') its *foster parent* in the spanning tree. In this case, sensor (i,j) becomes connected to the tree and it can route the data messages to the base station, but it does not send any connected(i,j) messages. (The reason for this last restriction is to prevent any subset of sensors from forming a directed cycle of foster parent relationship.)

When a sensor (i,j) is connected to the spanning tree via a foster parent (i',j'), and receives a connected(i'',j'') message from a sensor (i'',j''), then sensor (i,j) makes sensor (i'',j'') its parent or its foster parent (depending on whether (i'',j'') is a potential parent of (i,j)) in the spanning tree.

When a sensor (i,j) is connected to the spanning tree via a foster parent, but does not receive any connected message for a period of $3 * T$ seconds, it becomes disconnected from the tree and no longer forwards data messages to the base station.

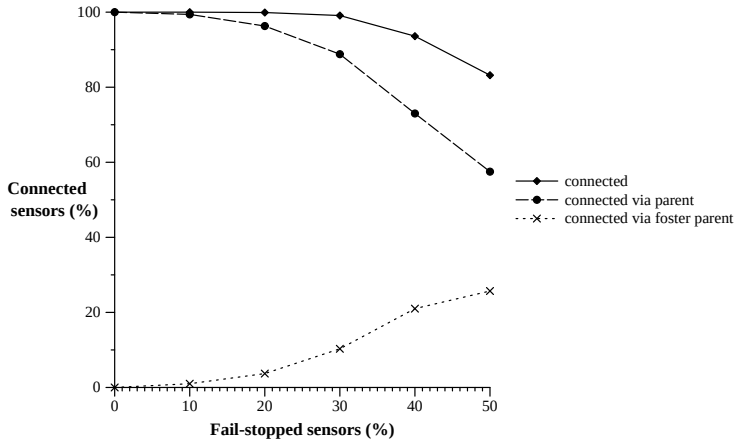


Fig. 2. Percentage of sensors that are connected to the tree vs. percentage of fail-stopped sensors

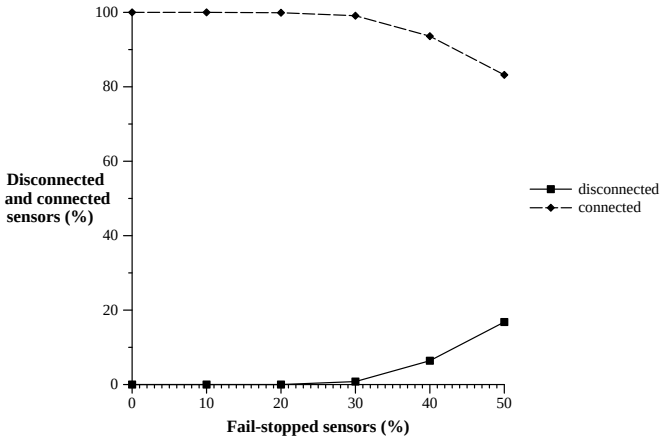


Fig. 3. Percentage of sensors that are disconnected from the tree vs. percentage of fail-stopped sensors

To evaluate the effectiveness of the foster parent extension, we considered a sensor network that has 100 sensors including the base station. This network is configured into a 10*10 logical grid with a hop size 2. We assumed that a percentage of the sensors in this network have fail-stopped and computed how many of the remaining sensors are still connected to the spanning tree via parents or via foster parents, and how many sensors become disconnected from the spanning tree. The results of these simulations are shown in Figures 2 and 3, and in Table I.

The good news from this study is that even if 50% of the sensors in the network have fail-stopped, about 84% of the remaining sensors remain connected to the tree. (Of those, 57% are

TABLE I

NUMBER OF FAIL-STOPPED SENSORS VS. NUMBER OF CONNECTED SENSORS IN 10*10 GRID WHEN H=2

# Fail-stopped sensors	# Connected sensors	# Disconnected sensors
0	100	0
10	89	1
20	79	1
30	68	2
40	56	4
50	42	8

connected via parents and 27% are connected via foster parents.) Only 16% of the remaining sensors become disconnected from the tree. These figures demonstrate that the foster parent extension is highly effective especially when a large fraction of the sensor in the network fail-stop.

VI. A BROADCAST PROTOCOL

As mentioned in Section I, a sensor network usually supports two communication patterns between the sensors in the network: unicast and broadcast. In the unicast pattern, any sensor in the network can send a message whose ultimate destination is the base station. In the broadcast pattern, the base station can send a message whose ultimate destination is each sensor in the network. So far we discussed how the logical grid routing protocol supports the unicast communication pattern. In this section, we show how to extend the logical grid routing protocol slightly to support the broadcast communication pattern.

Because each sensor sends a connected message every T seconds and because each connected message consists of 2 – 4 bytes, the broadcast message from the base station can be piggybacked on the sent connected message. Therefore, each connected message, sent by a sensor (i,j), becomes of the following form:

```
connected(i,j,seq,txt)
```

where seq is the sequence number of the broadcast message and txt is the text of the broadcast message (being piggybacked on the sent connected message).

After the base station piggybacks a broadcast message on a sent connected message, it

piggybacks the same message on the next $k-1$ connected messages. This means that the base station sends the same broadcast message k times, so that the probability that all the sensors in the network have received this message is very high. (We give an estimate of k below.)

This feature is implemented by providing the base station with a variable trb whose value is in the range $0..k$. When the base station piggybacks a new broadcast message on a sent connected message, trb is assigned the value k. Every time the base station times-out to send a connected message, the value of variable trb is decremented by one, and the last broadcast message is piggybacked on the sent connected message. When the value of variable trb becomes zero, the base station constructs a new broadcast message and starts to piggyback this message on the sent connected message and so on.

A specification of the base station is as follows:

```
sensor (0,0)              // base station

const T   : integer,      // avg interval to send next connected msg
      k   : integer       // time remaining to broadcast next msg

var   r   : 1..2*T - 1,   // random interval whose avg length is T
      trb : 0..k,         // time remaining to broadcast next msg, initially 0
      seq : integer,      // seq num of current broadcast msg, initially 0
      txt : integer       // text of current broadcast msg

begin
   time-out expires -> trb := max(trb-1,0);
                       if trb=0 -> txt := any;  // construct new broadcast msg
                                   seq := seq+1;
                                   trb := k
                       [] trb>0 -> skip
                       fi;
                       send connected(0,0,seq,txt);
                       r := rand; time-out after r

end
```

Each sensor (i,j), where $i \neq 0$ or $j \neq 0$, remembers the sequence number seq and the text txt of the last broadcast message it has received. When this sensor receives a connected(i',j',s,t) message, it compares seq with s. If seq is smaller than s, sensor (i,j) recognizes that the broadcast message that is being piggybacked on the received connected message is new. In this case, sensor

(i,j) accepts the broadcast message and updates its two variables seq and txt. If seq is at least s, then the sensor (i,j) recognizes that the broadcast message that is being piggybacked on the received connected message is old and ignores the broadcast message.

A specification of sensor (i,j), where $i \neq 0$ or $j \neq 0$, is as follows:

```
sensor (i,j)                        // a sensor (i,j) where i=!0 or j!=0

const P      : set of potential parents of sensor (i,j),
      T      : integer              // avg interval to send next connected msg

var   pid    : an element from P, // parent identifier
      trc    : 0..3,              // time to remain connected, initially 0
      r      : 1..2*T - 1,        // random interval whose avg length is T
      x      : 0..M-1,
      y      : 0..N-1,
      seq,s  : integer,           // seq num of current broadcast msg, initially 0
      txt,t  : integer            // text of current broadcast msg

begin
   rcv connected(x,y,s,t) ->
      if seq < s      ->  seq := s; txt := t
      [] seq >= s     ->  skip
      fi;
      if (x,y) in P    -> pid := (x,y);          // choose new parent
                          if trc=0 -> r := rand; time-out after r
                          [] trc>0 -> skip
                          fi; trc := 3
      [] !((x,y) in P) -> skip
      fi

[] time-out expires ->    trc := max(trc-1,0);
                          if trc>0 -> send connected(i,j,seq,txt);
                                      r := rand; time-out after r
                          [] trc=0 -> skip       // lose parent
                          fi
end
```

Next, we give an estimate of the parameter k mentioned above. The base station needs to keep on piggybacking the same broadcast message for a time period $k * T$ seconds before it piggybacks a new broadcast message on its sent connected messages. The period $k * T$ should be large enough to ensure that the current broadcast message is received by every sensor in the network. This means that the current broadcast message needs to make $\lceil \frac{(M-1)+(N-1)}{H} \rceil$ hops over

the logical grid. Thus, we have the relationship

$$k = \lceil \frac{(M-1)+(N-1)}{H} \rceil$$

As an example, consider a sensor network that is configured into a 10*10 logical grid whose hop size H is 2. In this case, the value of k is 9. If T is chosen to be 20 seconds in this network, then the base station can send a new broadcast message every $k * T = 9 * 20 = 180$ seconds, which is a reasonable broadcast rate for sensor networks.

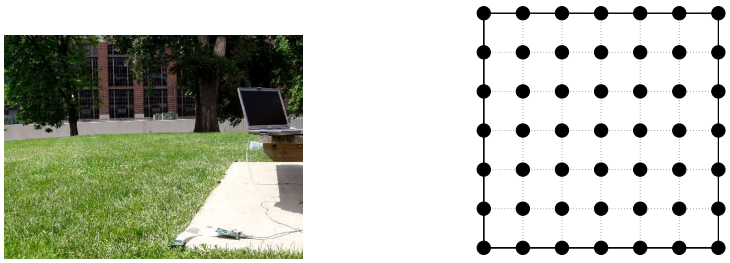
The sequence numbers of broadcast messages can have a limited range 0..$qmax$ where $qmax$ is any even positive integer. Each number in this range 0..$qmqx$ has $\frac{qmax}{2}$ numbers "larger" than it and $\frac{qmax}{2}$ numbers "smaller" than it. As an example, consider the case that the sequence numbers of broadcast messages are in the range of 0..4. In this case, 0 has two larger numbers 1 and 2 and two smaller numbers 3 and 4. Also, 1 has two larger numbers 2 and 3 and two smaller numbers 4 and 0 and so on.

VII. Experimental Results

We have applied the logical grid routing protocol to the DARPA Network Embedded Software Technology (NEST) field experiment "A Line in the Sand" [1], where about 80 MICA2 motes were deployed to monitor a field so that intruders (e.g., tanks, cars, and civilians) can be detected, classified, and tracked. This experiment successfully showed that the logical grid routing protocol is able to reliably route data messages from every sensor across the network and thus provides the foundation for precise target detection, classification, and tracking. Based on this experience, we have applied the logical grid routing protocol to the DARPA Extreme Scaling project (Exscal) field experiment [8], where about 1000 XSM motes and 45 Stargates (as the base stations) were deployed. This experiment also showed that the logical grid routing protocol can be used for large scale networks.

We implemented the logical grid routing protocol and a version of the distance vector routing protocol over the TinyOS platform[9]. We set up a testbed where 49 MICA2 motes[9] are deployed in a grass field (see Fig. 4(a)), forming a 7*7 grid (see Fig. 4(b)) with 5-feet separation

between neighboring grid points, and the base station (0,0) is the mote at the left-bottom corner of the grid.



(a) Testbed environment (b) Grid topology

Fig. 4. The network topology of the testbed

For our experiment of the logical grid routing protocol, we chose the hop size H to be 2 and T to be 20 seconds. To have a valid logical grid, the power level of each sensor was assigned 9 (out of the range 1..255). Based on this setup, we assigned each sensor in the testbed an identifier so that each sensor can compute the identifiers of its potential parents by using the algorithm in Section II. Note that the average number of hops from a sensor to the base station is around 3.3.

In each experiment, we used the traffic trace from a field experiment "A Line in the Sand"[1] to simulate the network load when events occur. The traffic trace corresponds to an event where each mote except the base station generates two data messages, and overall 96 data messages are generated. The cumulative distribution of the number of data messages that are generated by the sensors in the network during the event is shown as Fig. 5. Each result in figures and tables represents the average value over 10 runs of this trace.

The performance of a routing protocol can be evaluated by the following four metrics:

- *Total delivery ratio*: the ratio of the total number of unique data messages received by the base station to the total number of data messages generated by all sensors in the network.
- *Individual delivery ratio*: the ratio of the number of unique data messages received by the

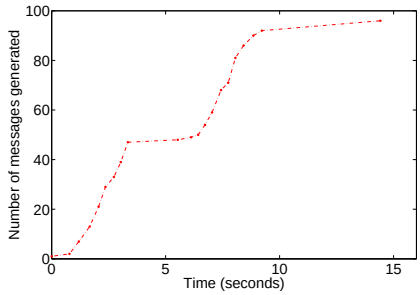


Fig. 5. The traffic distribution

base station from a particular sensor to the number of data messages generated by that particular sensor.

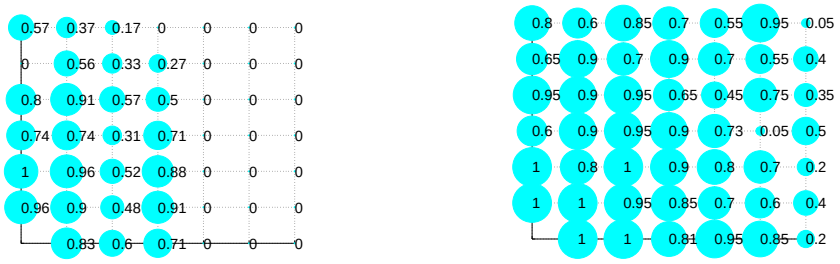
- *Delay*: the average time taken for a data message to be received by the base station after the data message is generated.
- *Goodput*: The number of unique data messages received by the base station divided by the interval between the time the first data message is generated and the time the base station receives the last data message. Note that the goodput reflects how fast data messages are pushed from the network to the base station. By definition, the optimal goodput for the trace is 6.66 messages/second, when the delay of each data message is 0 and all the data messages are received by the base station.

First, we ran experiments of the logical grid routing protocol and the distance vector routing protocol where each of the routing protocols uses the default TinyOS queue management component "QueuedSend"[9] and the maximum number of retransmissions of a data message is zero. The results are shown in Fig. 6, and Tables II and III.

TABLE II
PERFORMANCE OF THE DISTANCE VECTOR ROUTING WITH QUEUEDSEND

Total delivery ratio	Delay	Goodput
33.7%	0.11 seconds	2.76 messages/second

Fig. 6 shows the individual delivery ratio of each sensor in the the network. From Fig. 6, we



(a) Distance vector routing

(b) Logical grid routing

Fig. 6. Individual delivery ratios in the distance vector routing and in the logical grid routing with QueuedSend

observe that in the distance vector routing protocol, the individual delivery ratio of each sensor differs significantly with one another, and the total delivery ratio is only 33.7%.

On the other hand, in the logical grid routing protocol, the individual delivery ratio of each sensor reduces gradually as the number of hops from a sensor to the base station increases, and the total delivery ratio is 72%. The logical grid routing protocol provides relatively uniform delivery of data messages from different locations, since it balances the load over the network and avoids causing severe contention or congestion at certain network locations. Table III also shows the results of the logical grid routing protocol with QueuedSend when the maximum number of retransmissions of a data message is 1 and 2.

TABLE III

PERFORMANCE OF THE LOGICAL GRID ROUTING WITH "QUEUESEND"

	Total delivery ratio	Delay	Goodput
0 retransmission	72%	0.09 seconds	4.52 messages/second
1 retransmission	77.6%	0.11 seconds	4.83 messages/second
2 retransmission	81%	0.12 seconds	4.82 messages/second

Next, to further improve the delivery ratio, we developed a transport protocol RBC[10]. In RBC, lost messages are detected reliably and retransmitted at appropriate time without introducing much additional contention or congestion to the network. We ran experiments of

the logical grid routing protocol where the protocol uses RBC and the maximum number of retransmissions of a data message is 2, and the results are shown in Fig. 7 and Table IV.

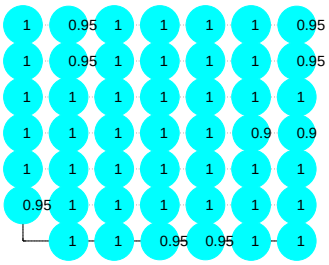


Fig. 7. Individual delivery ratios in the logical grid routing with RBC

From Fig. 7, we observe that the individual delivery ratio of each sensor is almost 100%, and the total delivery ratio is 98.8%. The delay increases since lost data messages are deferred in retransmission, but this delay does not affect the goodput of the protocol which might be more important to sensor network applications. The goodput reaches 6.45 messages/second that is very close to the optimal goodput 6.66 messages/second.

TABLE IV
PERFORMANCE OF THE LOGICAL GRID ROUTING WITH RBC

Total delivery ratio	Delay	Goodput
98.8%	1.2 seconds	6.45 messages/second

VIII. ADVANTAGES AND LIMITATIONS

The logical routing protocol has the following six advantages (over other routing protocols in sensor networks).

- Simplicity
- Load balancing
- Increasing the network lifetime
- Fast fault recovery

- Security without cryptography
- Supporting a broadcast protocol

Next, we discuss each of the advantages of the logical grid routing protocol.

i) Simplicity: The logical grid routing protocol is simple and so it consumes a small percentage of the resources of its network. In particular, the protocol requires that every sensor in the network sends only one routing message every 20 seconds, and stores no more than 10–15 bytes of routing information (which includes P, pid, trc, seq and txt). Also, each routing message has no more than 20 – 30 bytes including a broadcast message.

ii) Load balancing: The data items from the same source sensor are likely to follow different paths to the base station resulting in balancing the load over the network. This feature can also help for sensors to avoid severe congestion and reduce message collision, when a burst of sensing events occurs in the network.

iii) Increasing the network lifetime: The logical grid routing protocol requires that each sensor consumes a small amount of energy, and the sensors balance energy load across the network. Thus, the protocol can increase the network lifetime.

iv) Fast fault recovery: When the current parent of a sensor fail-stops, the sensor will replace this parent as soon as it receives a connected message from another potential parent. Thus, sensors in the network can recover quickly from the situation where their parents fail-stop.

v) Security without cryptography: In the distance vector routing protocol, an adversary sensor i can tempt many sensors to choose i to be their parents in the spanning tree by advertising a very small distance, say 1, to the base station. The adversary then can drop all the data messages that it receives from these sensors. However, in the logical grid routing protocol, each sensor has a set of potential parents in the spanning tree and keeps changing its parent whenever it receives a connected message from another potential parent. Thus, the adversary cannot make the sensor choose the adversary to be the parent of the sensor and then drop all data messages from that sensor, unless all the (legitimate) potential parents of the sensor fail-stop. The logical grid routing protocol provides this security feature without encrypting and decrypting connected messages.

vi) Supporting a broadcast protocol: Since each sensor sends connected messages periodically and connected messages consist of only 2 – 4 bytes, the broadcast messages from the base station can be piggybacked on the connected messages. The logical grid routing protocol is extended slightly to piggyback a broadcast message on the connected messages.

On the other hand, the logical grid routing protocol has the following four limitations.

- Initial setup requirement
- Limited communication patterns
- Limited connectivity
- No support for mobility

Next, we discuss each limitation of the protocol.

i) Initial setup requirement: The logical grid routing protocol needs to be set up. In particular, each sensor in the network needs to be assigned a distinct identifier that reflects the physical location of the sensor and a hop size H needs to be selected. Each sensor can use a GPS device or a localization service [11], [12], [2], specially for a large network. However, once this initial setup is over, the sensors can take all the advantages that the logical grid routing protocol provides.

ii) Limited communication patterns: The logical grid routing protocol does not support every communication pattern that can happen in a network. For example, two sensors in the network cannot exchange data messages unless one of them is the base station. Nevertheless, the protocol provides the two communication patterns that are most commonly used by sensor network applications.

iii) Limited connectivity: The logical grid routing protocol limits the connectivity of the sensors in a network such that sensors connected to the tree via foster parents do not send connected messages. Due to this limitation, some sensor in the network may not be connected to the tree. For example, in Fig. 8, sensor (4,4) is connected to the tree via a foster parent (3,4) when $H = 2$, and so does not send connected messages. Thus, sensor (4,6) cannot be connected to the tree. However, the probability that this case happens in the network is very small. We showed in Section V that even when 50% of sensors in the network are failed, 84% of the

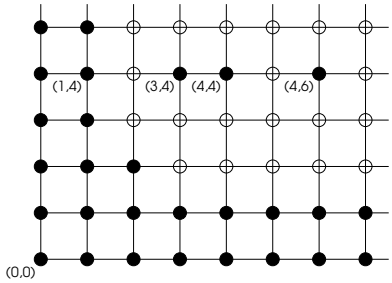


Fig. 8. A logical grid where a filled circle represents an alive sensor and a hollow circle represents a fail-stopped sensor

sensors still can be connected to the tree.

iv) No support for mobility: The sensors in the logical grid are assigned distinct identifiers such that their identifiers reflect their physical adjacencies. This cannot be maintained in mobile environments. However, in most sensor network applications, the sensors, specially the sensors that participate in routing, are stationary in the network.

IX. Related Work

Several routing protocols for sensor networks have been proposed and these protocols were categorized and classified in [13] and [14]. In [15], the characteristics of wireless communication that need to be considered for design of routing protocols were investigated.

In location-based routing protocols [5], each node utilizes its location, its neighbors' locations and a final destination location to forward messages to the destination. GPSR[16] uses greedy forwarding to forward a message to the neighbor that is geographically closest to the destination of the message, and uses perimeter forwarding when greed forwarding is not possible. GRID[17] and LAR[18] utilize location information to limit flooding in mobile ad hoc networks for route discovery, packet relay or route maintenance. GEAR[19] focuses on forwarding messages to a target region. GLS[20] uses geographic forwarding for a location service that tracks mobile node locations. In the logical grid routing protocol, each sensor computes its potential parents based on its logical grid identifier that reflects the (physical) location of the sensor, and chooses one of its potential parents to be its parent in the spanning tree.

Speed[7] combines a real-time protocol and a location-based routing protocol such that each node chooses the next hop among the neighbor nodes that are in “forwarding candidate set” of a message and also satisfy the desired relay speed. In [21], the authors investigated link selection strategies for a location-based routing protocol in lossy wireless sensor networks.

In [22], Woo et al investigated link quality estimation and neighbor management in dynamic and lossy sensor networks, and developed a version of the distance vector routing protocol, where each node selects a path with the minimal number of retransmissions based on link quality. In this approach, it is important to have stability in changing a parent, since each change is propagated over the network. In the logical grid routing protocol, changing a parent in a sensor does not affect any other sensors in the network.

In data-centric protocols such as Directed Diffusion [23], a sink floods a query over a network and sets up a path with a source. These query-based protocols may not work efficiently for applications where all nodes report events to a fixed base station. In [24], the combination of push-based strategy and pull-based strategy was studied for efficient information dissemination and gathering.

Multipath routing protocols have been proposed for load balancing and fault tolerance. Ganesan et al [25] utilize multiple paths to find an alternative path quickly when some node in the path between a sink and a source fail-stops, but there is overhead to maintain multiple paths. In [26], the algorithms of constrained random walks on random graphs were studied to construct multiple paths without any overhead and to provide load balancing over the network. In [27], the distance vector routing protocol is modified, for load balancing, such that each node distributes the traffic load over the nodes that have less or equal distance to the destination. In the logical grid routing protocol, sensors in the network can achieve load balancing and fast fault recovery without any overhead to maintain multiple paths and other information.

In [28], the convergecast flooding policies were proposed where a node rebroadcasts a message based on its gradient to the base station, or the information of its ancestors in the spanning tree whose root is the base station.

Trickle [29] propagates and maintains code over a network by “political gossip”. A node sends

metadata that shows the version of code the node has, only if the node has not received any metadata from any node in the network for a certain period. A node requests update or propagates update based on received metadata. In [30], nodes in the network first find an optimal target radius of the communication range that reduces energy consumption and the number of retransmissions, compute a connected dominating set, and then use this structure for broadcasting messages.

X. Concluding Remarks

In this paper, we presented the logical grid routing protocol that maintains an incoming spanning tree whose root is the base station. This tree is to be used to route data messages from any sensor to the base station in the network. We also extended the logical grid routing protocol slightly to piggyback a broadcast message on connected messages.

Our protocol is simple and so it consumes a small percentage of the resources of its network. In particular, the protocol requires that every sensor in the network sends only one routing message every 20 seconds and stores no more than 10 – 15 bytes of routing information. Also, each routing message has no more than 20 – 30 bytes. We showed by simulation that even if 50% of the sensors in a network fail-stop, 84% of the remaining sensors can still route data messages. The experimental results showed that the protocol delivers 72% – 99% of the messages to the base station under heavy and bursty traffic that generated 100 data messages per 15 seconds.

When the base station is located in the middle of a logical grid, the logical grid can be divided into four sub-grids. Then the sensors in each sub-grid execute the logical grid routing protocol to build a spanning tree whose root is the base station.

The logical grid routing protocol described in Section III can be easily modified to support multiple base stations. For example, each sensor can have two sets of potential parents, one set for the spanning tree whose root is the primary base station and the other set for the spanning tree whose root is the secondary base station. When the primary base station fail-stops, sensors in the network can route data messages to the secondary base station.

ACKNOWLEDGMENT

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Contract F33615-01-C-1901.

REFERENCES

[1] A. Arora, P. Dutta, and S. Bapat ut al, "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Computer Networks (Elsevier), Special Issue on Military Communications Systems and Technologies*, vol. 46, no. 5, pp. 605–634, December 2004.

[2] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *Proceedings of the ACM Second International COnference on Embedded Networked Sensor Systems (SenSys 04)*, Baltimore, MD,, November 2004, pp. 1–12.

[3] I. F. Akyildiz, W. Su, Y.Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks, Elsevier Science*, vol. 38, no. 4, pp. 393–422, 2002.

[4] A. Mainwaring, J. Polastre, R.D. Culler, and J. Anderson., "Wireless sensor networks for habitat monitoring," in *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, September 2002.

[5] I. Stojmenovic, "Position based routing in ad hoc networks," *IEEE Communications Magazine*, vol. 30, no. 7, pp. 128–134, 2002.

[6] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99)*, Seattle, Washington, August 1999.

[7] Tian He, John A. Stankovic, Chenyang Lu, and Tarek F. Abdelzaher, "Speed: A stateless protocol for real-time communication in sensor networks," in *Proceedings of the International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[8] "Exscal," http://www.cse.ohio-state.edu/exscal.

[9] Berkeley DARPA-NEST team, "Wireless embedded systems," *http://webs.cs.berkeley.edu/*.

[10] H. Zhang, A. Arora, Y.-R. Choi, and M. Gouda, "Reliable bursty convergecast in wireless sensor networks," Tech. Rep. CISRC-7/04-TR42, The Ohio State University, 2004.

[11] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones, "Training a wireless sensor network," *Mobile Networks and Applications*, vol. 10, no. 1-2, 2005.

[12] Jeffrey Hightower and Gaetano Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, August 2001.

[13] K. Akkaya and M. Younis, "A survey of routing protocols in wireless sensor networks," *Elsevier Ad Hoc Network Journal (to appear)*.

[14] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wireless Communications*.

[15] M. Haenggi, "Routing in ad hoc networks-a wireless perspective," in *Proceedings of the First International Conference on Broadband Networks (BroadNets'04)*, San Jose, CA, October 2004.

[16] B. Karp and H.T. Kung, "Greedy perimeter stateless routing for wireless networks," in *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August 2002, pp. 243–254.

[17] W.-H. Liao, Y.-C. Tseng, and J.-P. Sheu, "Grid: A fully location-aware routing protocol for mobile ad hoc networks," *Telecommunication Systems*, vol. 18, no. 1, pp. 37–60, 2001.

[18] Young-Bae Ko and Nitin H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," *Wireless Networks archive*, vol. 6, no. 4, pp. 307–321, 2000.

[19] Yan Yu, Ramesh Govindan, and Deborah Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," *UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023*, 2001.

[20] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris, "A scalable location service for geographic ad hoc routing," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, Boston, Massachusetts, August 2000, pp. 120–130.

[21] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[22] Alec Woo, Ternence Tony, and David Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of ACM SenSys*, Los Angelos, CA, 2003.

[23] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah 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.

[24] X. Liu, Q. Huang, and Y. Zhang, "Combs, needles, and haystacks: Balancing push and pull in large scale sensor networks," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[25] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review (MC2R)*, vol. 1, no. 2, 2002.

[26] S. D. Servetto and G. Barrenechea, "Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[27] Jorge Cobb and Mohamed Gouda, "Balanced routing," in *Proceedings of IEEE International Conference on Network Protocols(ICNP'97)*, 1997.

[28] Miklos Maroti, "The directed flood routing framework," *Institute Software Integrated Systems Vanderbilt University ISIS-04-052*, 2004.

[29] Philip Levis, Neil Patel, Scott Shenker, and David Culler, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor network," in *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004.

[30] M. Hauspie, A. Panier, and D. Simplot-Ryl, "Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks," in *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, Fort Lauderdale, USA, 2004.