JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 3, May 2006

Stabilization of Grid Routing in Sensor Networks

Young-ri Choi,* and Mohamed G. Gouda†
The University of Texas at Austin, Austin, TX 78712-0233
and
Hongwei Zhang,‡ and Anish Arora§
The Ohio State University, Columbus, OH 43210-1277

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 is stabilizing such that starting from any state, the protocol converges to a state where all the sensors and only the sensors that can be connected to the routing tree are connected to the tree. The convergence time of the protocol is proportional to the diameter of the sensor network. The routing protocol also has several other advantages over earlier protocols: overhead of the protocol is small, the protocol avoids unreliable long links to build a reliable routing tree, the protocol balances the load over the whole network, and it has nice fault-tolerance property. We have evaluated this protocol over a sensor network that consisted of about 50 MICA2 motes using a realistic traffic trace, and observed that the protocol delivers 72–99% 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 their neighbors 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] disaster monitoring,[2] and habitat monitoring.[3]

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

Received 16 September 2005; revision received 11 January 2006; accepted for publication 17 March 2006.

* Department of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin, TX 78712-0233, USA. yrchoi@cs.utexas.edu

† Department of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin, TX 78712-0233, USA. gouda@cs.utexas.edu

‡ Department of Computer Science and Engineering, The Ohio State University, 395 Dreese Laboratories, 2015 Neil Avenue Columbus, OH 43210-1277, USA. zhangho@cse.ohio-state.edu

§ Department of Computer Science and Engineering, The Ohio State University, 395 Dreese Laboratories, 2015 Neil Avenue Columbus, OH 43210-1277, USA. anish@cse.ohio-state.edu

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 parameters at the different sensors or reset the whole network. A routing tree whose root is the base station can be used to provide both communication patterns.

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 network resources. For example, sensors should not store a large routing table. Also, sensors should not send large routing messages frequently.

Second, sensors in a sensor network may start to execute the routing protocol from an illegitimate state. Also the sensors can be unreliable. For example, some sensors in the network may fail-stop when they run out of their energy or when they are physically damaged. Any routing protocol for sensor networks should be able to stabilize to a legitimate state, starting from any state, and also should be able to recover from the situation where a sizable fraction of the sensors fail-stop.[4]

Third, several previous experimental studies showed the existence of unreliable long links in sensor networks.[5–9] When sensors in a sensor network build a routing tree, these unreliable links should be avoided, since they result in poor message delivery. Not all long links are unreliable, but sensors cannot easily identify which long links are reliable and which are unreliable, especially when the traffic and the environment are changing dynamically. For this reason, distance vector routing protocols[10] may not work well in sensor networks, since these protocols take advantage of all long links but many of them are unreliable. One approach to avoid unreliable links is to estimate link quality dynamically using beacons.[6,7] Nevertheless, link quality is highly affected by environment, traffic pattern, and interference,[5,6,11] and so it is hard to estimate link quality precisely, especially under bursty traffic which is a common traffic pattern in sensor networks.[11]

In this paper, we present a routing protocol, called the logical grid routing (LGR) protocol, that overcomes the challenges of sensor networks. First, the protocol is simple and so it consumes a small percentage of the network resources. In particular, the protocol requires that every sensor in the network sends only one routing message that consists of 2–4 bytes every 20 seconds (or less frequently), and stores no more than 10–15 bytes of routing information. Second, the routing protocol is stabilizing such that starting from any state, the protocol converges to a state where all the sensors and only the sensors that can be connected to the routing tree are connected to the tree. We show that the convergence time of the protocol is proportional to the diameter of the sensor network. We also show that even if 50% of the sensors in a network fail-stop, 84% of the remaining sensors in the network can still route data messages. Third, we adopt a simple approach to avoid unreliable long links in the routing tree. In this approach, we use off-line experiments to estimate link quality, and use these results to identify reliable links in the routing tree. The experiments in our testbed showed that this simple approach works well in practice. Moreover, the LGR protocol has been used to successfully support reliable delivery of bursty traffic in a large scale sensor network.[12]

The routing tree in our protocol is an incoming spanning tree whose root is the base station. The routing 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 routing 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 LGR 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.

The first version of the LGR protocol is loop-free such that the protocol never forms any loops during the execution of the protocol. The protocol inherently achieves this loop-free property, since each sensor computes its potential parents based on its own identifier (that reflects its physical location in the network), and it chooses one of its potential parents to be its parent in the routing tree. The extended version of the LGR protocol is not loop-free, so it may form temporary loops when the protocol starts from an illegitimate state or when some sensors in the network fail-stop. Although this protocol is not loop-free, the foster parent extension is highly effective when all potential parents

of a sensor fail-stop, and the convergence time of this protocol is still proportional to the diameter of the sensor network.

The rest of the paper is organized as follows. In Section II, we present a model of the execution of a sensor network. In Section III, we discuss a logical grid and potential parents of sensors in the logical grid. We present the LGR protocol in Section IV and prove that this protocol is stabilizing in Section V. We present the extended protocol with foster parents in Section VI and prove that this protocol is stabilizing in Section VII. We show the experimental results of the LGR protocol in Section VIII. In Section IX, we discuss the advantages and limitations of the LGR protocol. We discuss related work in Section X, and finally make concluding remarks in Section XI.

II. Model of Sensor Network Execution

A sensor can be specified as a program that has global constants, local variables, one receiving action, and one timeout action. In general, a sensor is specified as Fig. 1.

A sensor u also has an implicit variable named *timer.u* to keep track of its timeout action. For every sensor u in the protocol, the value of *timer.u* is a non-negative integer in the range 0 .. *rmax*, for some specified *rmax* time units. Executing the receiving action of sensor u causes u to update its own local variables. It may also cause u to execute the statement "timeout-after <expression>" which causes the timeout of u to expire after k time units, where k is the current value of <expression> and $k > 0$. Note that executing the receiving action of sensor u does not cause u to send any message.

The timeout action in a sensor u is executed when *timer.u* has a value of zero. Executing the timeout action of sensor u causes u to update its local variables, and to send at most one message. It also causes u to execute the statement "timeout-after <expression>". If sensor u sends a message, then each neighbor v of sensor u may receive the message. We assume that neighbor relation is symmetric. Thus, if sensor u can receive a message sent by v, sensor v can receive a message sent by u. The timeout action of sensor u is of the following form:

```
timeout-expires -> <update local variables of u>;
                   <send at most one message>;
                   <execute timeout-after <expression>>
```

The execution of a protocol proceeds as follows. First, the smallest value of *timer.u* is identified from all sensors in the protocol. Second, for every sensor u, the value of *timer.u* is decreased by the identified smallest value. Then there exists at least one sensor u whose value of *timer.u* is zero in the protocol. Third, a sensor u is selected arbitrarily from all sensors whose timer variables have zero values, and executes its timeout action. If sensor u sends a message during the execution of its timeout action, then each neighbor v of sensor u may receive the message at the exactly same instant. Each sensor v that receives the message sent by sensor u executes its receiving action. Note that at this point, every sensor in the protocol has an updated timer value. This process is repeated over and over during the execution of the protocol.

If a sensor u receives a message <msg>, then u executes the following receiving action.

```
rcv <msg> -> <update local variables of u>;
             <may execute timeout-after <expression>>
```

```
sensor <sensor name>

const <const name> : <const type>, ... , <const name> : <const type>
var   <var name> : <var type>, ... , <var name> : <var type>

begin
   rcv <msg>        -> <action statements>          // receiving action
[] timeout-expires  -> <action statements>          // timeout action
end
```

Fig. 1 Sensor specification.

III. Logical Grid and Potential Parents

In this section, we first describe a logical grid and potential parents of sensors in the logical grid, and present an algorithm to compute potential parents for each sensor. We then discuss one of the methods to build a logical grid.

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 $i = 0 \cdots M - 1$ and $j = 0 \cdots N - 1$. The base station in this network is sensor (0, 0).

(To keep our presentation simple, we choose the base station to be the grid point (0, 0). In practice, 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.)

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 neighbor (i', j'), where $i \geq i'$ and $j \geq j'$, that is closer towards (0, 0) than (i, j). The transmission of the data item from sensor (i, j) to sensor (i', j') is called *one hop*. The "physical" distance between sensors (i, j) and (i', j') should be small enough in order not to create a long link, but should be large enough in order to reduce the number of hops until the data item reaches its final destination.

Roughly, the requirement that sensor (i, j) forwards its data item to a neighbor (i', j') can be stated 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 routing tree whose root is the base station (0, 0). Thus, sensor (i, j) can have up to $H + 1$ potential parents, sensors $(i, j - H)$, $(i - 1, j - H + 1) \cdots (i - H + 1, j - 1)$, and $(i - H, j)$. Note that the above characterization of potential parents may not be valid for sensors on or near the boundary of the grid. We describe below an algorithm to compute potential parents of each sensor including those on or near the boundary of the grid.

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. 2, 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 algorithm in Fig. 3 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.

Next, we discuss one of the methods to build a logical grid and compute the value of H. (Similar methods to build a logical grid for the case $H = 1$ can be found in.[13,14]) We divide a network area into grid squares where each size is $d * d$. In this network, a sensor is assigned a logical identifier (i, j), if the sensor is deployed at the grid square (i, j). The sensor acts as if it is deployed at the grid point (i, j), that is the central point of the grid square, of the logical grid. For example, Fig. 4 shows a network area that consists of three grid squares. In Fig. 4, a dashed square represents each grid square, and solid lines represent the imposed logical grid for the network area. Note that the value of d needs to be selected such that at least one sensor is deployed at each grid square with a high probability.

We assume that each sensor knows its physical location (x, y) using GPS or localization services in[15,16] and the side length of a grid square d. We also assume that the reliable transmission range of sensors is R. Using its physical

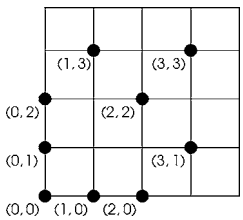


Fig. 2 A 5 * 5 logical grid where the hop size is 2.

```
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
```

Fig. 3 Algorithm to compute potential parents.

location and d, each sensor decides its logical identifier (i, j). (In this work, we assume that only one sensor is deployed at each grid square. If two or more sensors are deployed at the same grid square, only one sensor among them can be selected and stay awake to save energy by using an energy saving protocol such as.[13,17])

In order to establish that the imposed logical grid and the selected H are valid, each sensor should be able to "reliably" exchange data messages with each of its potential parents. In other words, R should be larger than the maximum distance between a sensor and its potential parents. Therefore, we need the following condition.

$$R \geq \sqrt{d^2 + ((H+1) * d)^2} \quad \text{i.e.} \quad R \geq d * \sqrt{H^2 + 2H + 2}$$

For example, consider the case $H = 2$. In this case, a sensor (i, j) has the maximum distance with its potential parent $(i-2, j)$ (or $(i, j-2)$). The maximum distance between two sensors deployed at grid squares (i, j) and $(i-2, j)$ is computed as $d * \sqrt{10}$, as shown in Fig. 4. Thus, R should be at least $d * \sqrt{10}$.

We perform off-line experiments to find the reliable transmission range R of sensors, and then we select the largest integer H that satisfies the above condition. Note that we need to conservatively measure R so that a sensor can communicate with each of its potential parents reliably, even under heavy interfering traffic.

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

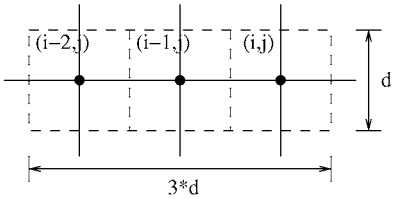


Fig. 4 Network area.

IV. The Logical Grid Routing Protocol

The purpose of the LGR protocol is to build and maintain a routing tree whose root is the base station (0, 0). Each sensor (i, j) chooses one of its potential parents to be its parent in this tree.

Initially, only the base station (0, 0) is in the routing tree. The base station periodically sends a message of the form, connected (0, 0), every random period whose length is chosen uniformly from the range *rmin .. rmax*, where the average length of the range *rmin .. rmax* is T time units. (This random period is to reduce the probability of connected message collision.)

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 random period whose length is chosen uniformly from the range *rmin .. rmax*.

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 random period whose length is chosen uniformly from the range *rmin .. rmax*.

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 time period of $tmax * T$ time units, sensor (i, j) concludes that it is no longer connected to the tree and stops sending connected (i, j) messages. (We discuss how to choose a value of *tmax* below.) 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.

A specification for the base station is given in Fig. 5.

We mentioned earlier that when a sensor does not receive a connected message from any of its potential parents for the time period of $tmax * T$ time units, the sensor recognizes that it is no longer connected to the routing 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 .. *tmax*. When sensor (i, j) receives a connected message from any of its potential parents, *trc* is assigned the value *tmax*. Every time sensor (i, j) times-out to send a connected message, the value of variable *trc* is decremented by one. When the value of variable *trc* becomes zero, sensor (i, j) recognizes that it is no longer connected to the routing tree.

During the execution of the protocol, a sensor u may consecutively choose small random periods for some number of times, and some of connected messages sent by the potential parents of sensor u may be lost. Nonetheless, we assume that sensor u is truly disconnected from the routing tree if the value of *trc* in u becomes zero. In order to establish that our assumption is valid, we need to assign *tmax* a reasonably large value that ensures the following with a high probability: if any potential parent v of a sensor u sends a connected message periodically, then sensor u is guaranteed to receive a connected message sent by v before the value of *trc* in u becomes zero. (Note that this value of *tmax* can be estimated based on the values of *rmin* and *rmax*, and the probability of message loss in the network.) From now on, we assume that a reasonable value that makes our assumption valid is assigned to *tmax*.

A specification for sensor (i, j), where $i \neq 0$ or $j \neq 0$, is given in Fig. 6.

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

const rmin,rmax : integer       // min and max interval whose avg is T
var   r         : rmin..rmax    // random interval to send next connected msg

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

Fig. 5 Specification of sensor (0, 0).

```
sensor (i,j)           // a sensor (i,j) in an M*N grid where i=!0 or j!=0

const P         : set of potential parents of sensor (i,j),
      rmin,rmax : integer,           // min and max interval whose avg is T
      tmax      : integer            // max time to be connected

var   pid       : an element from P, // parent identifier
      trc       : 0..tmax,           // time to remain connected
      r         : rmin..rmax,        // random interval to send next connected msg
      x         : 0..M-1,
      y         : 0..N-1

begin
   rcv connected(x,y) ->  if (x,y) in P    -> pid := (x,y);  // choose new parent
                             trc := tmax
                          [] !((x,y) in P) -> skip
                          fi

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

Fig. 6 Specification of sensor (i, j).

Note that in the LGR 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 real-time of when they were generated.)

Next, we specify how this protocol is used to route data messages to the base station (0, 0). When a sensor (i, j), other than the base station, has a data message to route to the base station, sensor (i, j) first checks whether or not it is connected to the routing 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. In this case, sensor (i, j) sends the message after attaching its parent identifier pid to the message. Otherwise, sensor (i, j) recognizes that it is not connected to the tree and drops the message.

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 routing 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.

V. Stabilization of the Protocol

In this section, we sketch a proof that the protocol in Section IV is self-stabilizing. This proof is based on the following four assumptions.

A1) Each sensor in the protocol is either up or down. Up sensors execute the protocol as desired, while down sensors just fail-stop. Moreover, whether a sensor is up or down does not change along any execution of the protocol.

A2) For every two distinct up sensors u and v, if u is a neighbor of v, or if there exists a third sensor w that is a neighbor for both u and v, then $timer.u$ and $timer.v$ have distinct values. (Note that this assumption is

probabilistically maintained by choosing the difference $rmax - rmin$ to be large relative to the number of neighbors of a sensor.)

A3) For every up sensor u, if the value of *trc* in sensor u is 1, then sensor u chooses a value for *timer.u* such that the value of *timer.u* is larger than the value of *timer.v* for any up potential parent v of sensor u.

A4) For every up sensor u, if the value of *trc* in sensor u is 0 or 1, and any potential parent of sensor u sends a connected message, then sensor u receives the connected message.

(Note that assumptions A3 and A4 ensure that if any potential parent v of a sensor u sends a connected message periodically, then sensor u is guaranteed to receive a connected message sent by v before the value of *trc* in u becomes 0.)

In the protocol, a *state* of a sensor u is defined by the value of variable *trc* and the value of variable *timer.u* in sensor u, and the state of each up potential parent v of sensor u. Sensor u is in one of the following two states: (We use the notation <var>.u to denote the value of variable <var> in sensor u.)

i) Sensor u is in a *c-state* iff u is the base station, or at this state
($trc.u > 1$ and there exists an up potential parent v of u where v is in a c-state) or
($trc.u = 1$ and there exists an up potential parent v of u where v is in a c-state and $timer.u > timer.v$)

ii) If sensor u is not in a c-state, then sensor u is in a *d-state*.

A *configuration* of the protocol is defined by the state for each up sensor in the protocol.

Regardless of the initial configuration of the protocol, from the grid size $M * N$, the hop size H, and the set of up sensors in the grid, we can compute the *status* (whether reachable or unreachable) of each up sensor in the protocol. The status of an up sensor u is *reachable* iff sensor u is the base station (0, 0), or the protocol has an up sensor v such that v is an element of $P.u$ and the status of v is reachable. Otherwise, the status of sensor u is *unreachable*. Note that a down sensor has no status, and if the base station is up, then its status is always reachable.

A configuration of the protocol is *legitimate* iff the following two conditions hold for every up sensor u in the protocol.

If sensor u is reachable, sensor u is in a c-state in the protocol configuration.

If sensor u is unreachable, sensor u is in a d-state and $trc.u = 0$ in the protocol configuration.

The protocol is *self-stabilizing* iff it satisfies the following two conditions.[18]

i) *Closure*: Starting from any legitimate configuration, the execution of any action in any sensor in the protocol yields a legitimate configuration.

ii) *Convergence*: Starting from any illegitimate configuration, the protocol is guaranteed to reach a legitimate configuration.

Lemma 1. *Starting from any legitimate configuration, for every reachable sensor u, the execution of any action in sensor u yields a legitimate configuration.*

Proof: The protocol has two cases to consider. In the first case, the executed action is the receiving action in sensor u. In this case, if sensor u receives a connected message from one of its potential parents, *trc.u* is set to *tmax*. Otherwise, sensor u drops the message and does nothing. Therefore, the receiving action in u yields a legitimate configuration. In the second case, the executed action is the timeout action in sensor u. In this case, sensor u first decreases *trc.u* by one. However, since u is reachable, there exists at least one up sensor v such that v is an element of $P.u$ and the status of v is reachable. Thus, u will receive a connected message from a sensor that is an element of $P.u$ before *trc.u* becomes 0, and so *trc.u* remains to be bigger than 0 (by assumptions A3 and A4). Second, u sends a connected message (since $trc.u > 0$). If there is a sensor v where $u \in P.v$ and $trc.v = 1$, then v receives the connected message sent by u and sets *trc.v* to *tmax* (by assumption A4). Third, u chooses a random value for *timer.u*. If $trc.u = 1$, u chooses a value for *timer.u* according to assumption A3, and so u remains in a c-state. Therefore, the timeout action in u yields a legitimate configuration.

Lemma 2. *Starting from any legitimate configuration, for every unreachable sensor u, the execution of any action in sensor u yields a legitimate configuration.*

Proof: The protocol has two cases to consider. In the first case, the executed action is the receiving action in sensor u. In this case, since sensor u is unreachable, the protocol has no up sensor v such that v is an element of $P.u$ and

the status of v is reachable. Thus, sensor u always drops the received message, and does nothing. Therefore, the receiving action in u yields a legitimate configuration. In the second case, the executed action is the timeout action in sensor u. In this case, *trc.u* remains 0 by executing the statement "max(*trc*-1, 0)". Therefore, the receiving action in u yields a legitimate configuration.

Theorem 1. (*Closure*) *Starting from any legitimate configuration, the protocol is guaranteed to maintain a legitimate configuration.*

Proof: The proof follows from the proofs of Lemmas 1 and 2.

The following definitions are useful to prove the convergence of the protocol. We define the *connection graph C* of the protocol as a directed graph that satisfies the following two conditions. First, each node in graph C represents a distinct sensor in the set of reachable sensors in the protocol. Second, each directed edge (u, v) from a node u to a node v in graph C indicates that sensor v is a potential parent of sensor u. We define the height of a connection graph C as the maximum number of edges in any path from any node to the base station in graph C. Let the height of graph C be K.

We define the *disconnection graph D* of the protocol as a directed graph that satisfies the following two conditions. First, each node in graph D represents a distinct sensor in the set of unreachable sensors in the protocol. Second, each directed edge (u, v) from a node u to a node v in graph D indicates that sensor v is a potential parent of sensor u. We define the height of a disconnection graph D as the maximum number of nodes in any path from any node to another node in graph D. Let the height of graph D be L.

Next, we show that starting from any illegitimate configuration, our protocol is guaranteed to reach a legitimate configuration within a finite time period.

Lemma 3. *Starting from any illegitimate configuration, every reachable sensor u is guaranteed to be in a c-state within* $K * tmax * rmax$ *time units.*

Proof: Let the base station be node u_0 in the connection graph C of the protocol. If u_0 is up, u_0 sends a connected message periodically. (Note that if u_0 is down, the connection graph of the protocol does not exist.) Thus, for every node u_1 where there exists edge (u_1, u_0) in graph C, u_1 is guaranteed to receive at least one connected message from u_0, and reach a c-state by setting *trc.u* to *tmax* within $tmax * rmax$ time units (by assumptions A3 and A4). Moreover, u_1 is guaranteed to start sending a connected message periodically within $tmax * rmax$ time units. Assume that for every node u_{K-1} where there exist edges (u_i, u_{i-1}), $1 \leq i \leq K-1$, in graph C, u_{K-1} is guaranteed to reach a c-state and start sending a connected message periodically within $(K-1) * tmax * rmax$ time units. For every node u_K where there exist edges (u_i, u_{i-1}), $1 \leq i \leq K$, in graph C, u_K is guaranteed to receive at least one connected message from any u_{K-1}, and reach a c-state within $K * tmax * rmax$ time units (by assumptions A3 and A4). Therefore, every node in graph C reaches a c-state within $K * tmax * rmax$ time units, if the height of graph C is K.

Lemma 4. *Starting from any illegitimate configuration, for every unreachable sensor u, it is guaranteed that sensor u is in a d-state and* $trc.u = 0$ *within* $L * tmax * rmax$ *time units.*

Proof: Every unreachable sensor u is always in a d-state. Therefore, we only need to show that *trc.u* becomes 0 within a finite time period. In graph D, there exists at least one node that has no outgoing edge, since all nodes in graph D are unreachable. For every node u_0 where there exists no outgoing edge from u_0 in graph D, u_0 will not receive any connected message from a node v where there exists edge (u_0, v) in graph D. Thus, it is guaranteed that $trc.u_0$ becomes 0 within $tmax * rmax$ time units. Assume that for every node u_{L-2} where there exist edges (u_i, u_{i-1}), $1 \leq i \leq L-2$, in graph D, it is guaranteed that $trc.u_{L-2}$ becomes 0 within $(L-1) * tmax * rmax$ time units. Since u_{L-2} does not send a connected message, for every node u_{L-1} where there exist edges (u_i, u_{i-1}), $1 \leq i \leq L-1$, in graph D, u_{L-1} will not receive any connected message from any u_{L-2}. Thus, it is guaranteed that $trc.u_{L-1}$ becomes 0 within $L * tmax * rmax$ time units. Therefore, for every node u in graph D, it is guaranteed that u is in a d-state and $trc.u = 0$ within $L * tmax * rmax$ time units, if the height of graph D is L.

Theorem 2. (*Convergence*) *Starting from any illegitimate configuration, the protocol is guaranteed to reach a legitimate configuration within max* ($K * tmax * rmax$, $L * tmax * rmax$) *time units.*

Proof: The proof follows from the proofs of Lemmas 3 and 4.

VI. The Routing Protocol with Foster Parents

According to the LGR protocol in Section IV, a sensor has a parent in the routing 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 up and connected to the routing 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 routing tree.

To solve this problem, we extend the LGR protocol such that a sensor remains connected to the routing tree even if all its potential parents have fail-stopped as follows. 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 routing 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 relationships.)

When a sensor (i, j) is connected to the routing 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 routing tree.

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

A specification for sensor (i, j), where $i \neq 0$ or $j \neq 0$, is given in Fig. 7. (Note that a specification for the base station is identical to the one in Section IV.)

```
sensor (i,j)           // a sensor (i,j) in an M*N grid where i=!0 or j!=0

const P         : set of potential parents of sensor (i,j),
      rmin,rmax : integer,           // min and max interval whose avg is T
      tmax      : integer

var   pid       : a sensor (i',j') where i!=i' or j!=j',  // parent identifier
      trc       : 0..tmax,           // time to remain connected
      r         : rmin..rmax,        // random interval to send next connected msg
      x         : 0..M-1,
      y         : 0..N-1

begin
   rcv connected(x,y) -> if ((x,y) in P) or (trc=0) or (trc>0 and !(pid in P)) ->
                            pid := (x,y);         // choose new parent
                            trc := tmax
                         [] !((x,y) in P) and (trc>0) and (pid in P) -> skip
                         fi

[] time-out expires ->   trc := max(trc-1,0);
                         if trc>0 and (pid in P) -> send connected(i,j);
                         [] trc=0 or !(pid in P) -> skip
                         fi; r := rand; time-out after r
end
```

Fig. 7 Specification of sensor (i, j).

VII. Stabilization of the Protocol with Foster Parents

In this section, we sketch a proof that the protocol in Section VI is self-stabilizing. This proof is based on assumptions A1 and A2 discussed in Section V, and the following two assumptions.

B3) For every up sensor u, if the value of *trc* in sensor u is 1, then sensor u chooses a value for *timer.u* such that the value of *timer.u* is larger than the value of $timer.v$ for any up neighbor v of sensor u.

B4) For every up sensor u, if the value of *trc* in sensor u is 0 or 1, and any neighbor of sensor u sends a connected message, then sensor u receives the connected message.

Note that the proof in this section is similar to that in Section V. However, the proof in this section needs to consider the cases where some sensors in the protocol have foster parents.

In the protocol, a *state* of a sensor u is defined by the value of variable pid, the value of variable *trc*, and the value of variable *timer.u* in sensor u, and the state of each up neighbor v of sensor u. Sensor u is in one of the following three states:

i) Sensor u is in a *c-state* iff u is the base station, or at this state
$pid.u \in P.u$ and (($trc.u > 1$ and there exists an up potential parent v of u where v is in a c-state) or
($trc.u = 1$ and there exists an up potential parent v of u where v is in a c-state and $timer.u > timer.v$))

ii) Sensor u is in a *j-state* iff at this state
$pid.u \notin P.u$ and (($trc.u > 1$ and there exists an up neighbor v of u where v is in a c-state) or
($trc.u = 1$ and there exists an up neighbor v of u where v is in a c-state and $timer.u > timer.v$))

iii) If sensor u is neither in a c-state nor in a j-state, then sensor u is in a *d-state*.

Regardless of the initial configuration of the protocol, from the grid size $M * N$, the hop size H, and the set of up sensors in the grid, we can compute the *status* (whether reachable, weakly reachable, or unreachable) of each up sensor in the protocol. The status of an up sensor u is *reachable* iff sensor u is the base station (0, 0), or the protocol has an up sensor v such that v is an element of $P.u$ and the status of v is reachable. The status of sensor u is *weakly reachable* iff sensor u is not reachable, and the protocol has an up sensor v such that v is an element of $Ngh.u$ and the status of v is reachable, where $Ngh.u$ is the set of up sensors that are neighbors of sensor u and are not elements of $P.u$. Otherwise, the status of sensor u is *unreachable*.

A configuration of the protocol is *legitimate* iff the following three conditions hold for every up sensor u in the protocol.

If sensor u is reachable, sensor u is in a c-state in the protocol configuration.
If sensor u is weakly reachable, sensor u is in a j-state in the protocol configuration.
If sensor u is unreachable, sensor u is in a d-state and $trc.u = 0$ in the protocol configuration.

Lemma 5. *Starting from any legitimate configuration, for every reachable sensor u, the execution of any action in sensor u yields a legitimate configuration.*

Lemma 6. *Starting from any legitimate configuration, for every weakly reachable sensor u, the execution of any action in sensor u yields a legitimate configuration.*

Lemma 7. *Starting from any legitimate configuration, for every unreachable sensor u, the execution of any action in sensor u yields a legitimate configuration.*

The proofs of Lemmas 5, 6, and 7 are similar to those of Lemmas 1 and 2.

Theorem 3. (*Closure*) *Starting from any legitimate configuration, the protocol is guaranteed to maintain a legitimate configuration.*

Proof: The proof follows from the proofs of Lemmas 5, 6, and 7.

We define the connection graph C of the protocol and the height of the connection graph, K, as they are defined in Section V. We also define the disconnection graph D of the protocol and the height of the disconnection graph,

L, as they are defined in Section V. Next, we show that starting from any illegitimate configuration, our protocol is guaranteed to reach a legitimate configuration within a finite time period.

Lemma 8. *Starting from any illegitimate configuration, every reachable sensor u is guaranteed to be in a c-state within* $K * tmax * rmax$ *time units.*

Proof: This proof is similar to that of Lemma 3.

Lemma 9. *Starting from any illegitimate configuration, every weakly reachable sensor u is guaranteed to be in a j-state within* $(K + 1) * tmax * rmax$ *time units.*

Proof: By Lemma 8, every reachable sensor is guaranteed to reach a c-state, and start sending a connected message periodically within $K * tmax * rmax$ time units. Since sensor u is weakly reachable, there exists at least one up sensor v such that v is an element of $Ngh.u$, and the status of v is reachable. Thus, every sensor u is guaranteed to receive at least one connected message from a sensor that is an element of $Ngh.u$, and reach a j-state within $(K + 1) * tmax * rmax$ time units (by assumptions B3 and B4).

Lemma 10. *Starting from any illegitimate configuration, for every unreachable sensor u, it is guaranteed that sensor u is in a d-state and* $trc.u = 0$ *within* $(L + 1) * tmax * rmax$ *time units.*

Proof: Every unreachable sensor u is always in a d-state. Therefore, we only need to show that $trc.u$ becomes 0 within a finite time period. For every node u_0 where there exists no outgoing edge from u_0 in graph D, u_0 will not receive any connected message from a node v where there exists edge (u_0, v) in graph D. Thus, it is guaranteed that if $trc.u_0 > 0$, $pid.u_0$ is not an element of $P.u_0$ within $tmax * rmax$ time units, and also $pid.u_0$ cannot be an element of $P.u_0$ from this point. Similar to the proof of Lemma 4, we can prove that for every node u in graph D, it is guaranteed that if $trc.u > 0$, $pid.u$ is not an element of $P.u$ within $L * tmax * rmax$ time units, and also $pid.u$ cannot be an element of $P.u$ from this point. Thus, every node u in graph D will not send a connected message, and so will not receive any connected message from any node in graph D. Also node u cannot receive any connected message from a node that is not in graph D. Thus, for every node u in graph D, it is guaranteed that u is in a d-state and $trc.u = 0$ within $(L + 1) * tmax * rmax$ time units, if the height of graph D is L.

Theorem 4. (*Convergence*) *Starting from any illegitimate configuration, the protocol is guaranteed to reach a legitimate configuration within max* $((K + 1) * tmax * rmax, (L + 1) * tmax * rmax)$ *time units.*

Proof: The proof follows from the proofs of Lemmas 8, 9, and 10.

VIII. Simulation and Experimental Results

In this section, we first show the effectiveness of the foster parent extension by simulation, and then discuss the experimental results of the LGR protocol.

Simulation results We evaluated the effectiveness of the foster parent extension by simulation. We considered a sensor network that 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 we computed how many of the remaining sensors are still connected to the routing tree via parents or via foster parents, and how many sensors become disconnected from the routing tree. The results of these simulations are shown in Figs. 8 and 9, and in Table 1, where each result represents the average value over 100 simulations.

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 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 sensors in the network fail-stop.

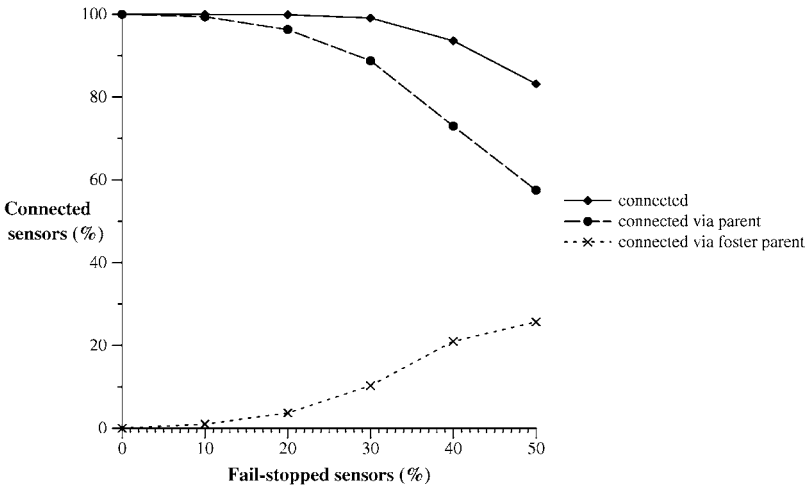


Fig. 8 Percentage of sensors that are connected to the tree vs. percentage of fail-stopped sensors.

Experimental results The LGR protocol has been used for the field experiment "A Line in the Sand",[1] where around 100 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 showed that the LGR protocol is able to reliably route data messages from every sensor across the network to the base station and thus provides the foundation for precise target detection, classification, and tracking. The LGR protocol also has been applied to the large scale field experiment "ExScal",[12] where about 1000 XSMs (for eXtreme Scale Motes) and 200 XSSs (for eXtreme Scale Stargates) were deployed to detect and track intruders. In this experiment, again the LGR protocol has been able to successfully provide reliable message delivery with the delay less than 2 seconds.

To study in more depth the property of the LGR protocol in forwarding packets from different locations, we set up a testbed where 49 MICA2 motes[19] are deployed in a grass field (see Fig. 10(a)), forming a $7 * 7$ grid (see Fig. 10(b))

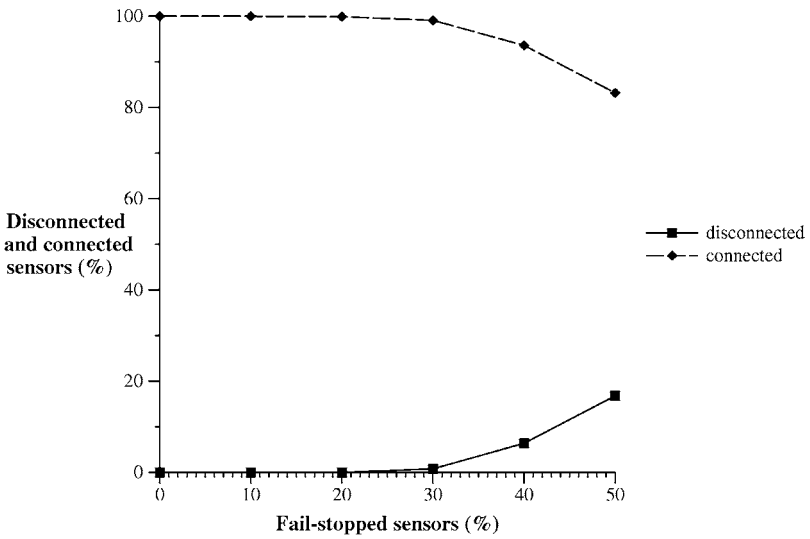


Fig. 9 Percentage of sensors that are disconnected from the tree vs. percentage of fail-stopped sensors.

Table 1 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

with a 5-feet separation between neighboring grid points. The base station (0, 0) is the mote at the left-bottom corner of the grid.

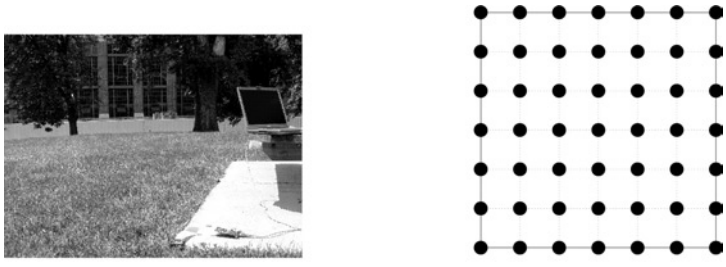
For our experiments, we chose the hop size H to be 2 and T to be 20 seconds. To have a valid logical grid, the power level of each sensor was assigned 9 (out of the range 1..255). Based on this setup, we assigned each sensor in the testbed an identifier so that each sensor can compute the identifiers of its potential parents by using the algorithm in Section III. In this testbed, 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 the 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 whose interval is between 3 and 4.5 seconds, 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. 11. Each performance result discussed in this section represents the average value over 10 runs of this trace.

We evaluate the performance of a routing protocol 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 (unique) data messages generated by all sensors in the network.
- *Individual delivery ratio*: the ratio of the number of unique data messages received by the 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 last data message is received by the base station. Note that the goodput reflects how fast data messages are pushed from the network to the base station.

First, we ran experiments of the LGR protocol with the default TinyOS queue management component "Queued-Send" which simply retransmits a message up to a certain number of times until the acknowledgment of the message is received.[20] Fig. 12 shows the individual delivery ratio when the maximum number of per hop retransmissions is 0 (that is, no retransmission). 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 LGR protocol provides reliable



(a) Testbed environment (b) Grid topology

Fig. 10 The network topology of the testbed.

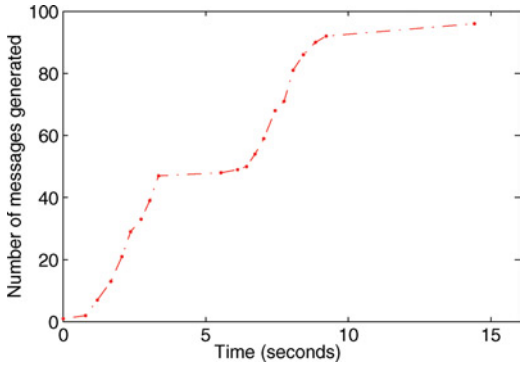


Fig. 11 The traffic distribution.

uniform delivery of data messages from different locations. This is because the LGR protocol avoids unreliable links to build a reliable routing tree. Also it balances the load over the network and avoids causing severe contention or congestion at certain network locations. Table 2 also shows the results of the LGR protocol with QueuedSend when the maximum number of per hop retransmissions is up to 2.

To further improve the delivery ratio, we used the LGR protocol with a transport protocol RBC developed in.[21] RBC replicates the acknowledgement for a received message using a window-less block acknowledgement scheme, and schedules message retransmissions to alleviate contention caused by them. Thus, 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 LGR protocol with RBC where the maximum number of per hop retransmissions is 2, and the results are shown in Fig. 13 and Table 3. From Fig. 13, 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. However, this delay does not decrease the goodput which is more important to sensor network applications. Moreover, the delay is good enough for some typical sensor network applications, such as intrusion detection and tracking.[1,12] The goodput reaches 6.45 messages/second. Note that the optimal goodput for

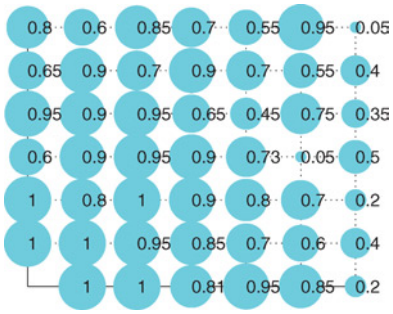


Fig. 12 Individual delivery ratios of LGR with QueuedSend.

Table 2 Performance of LGR with QueuedSend.

	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

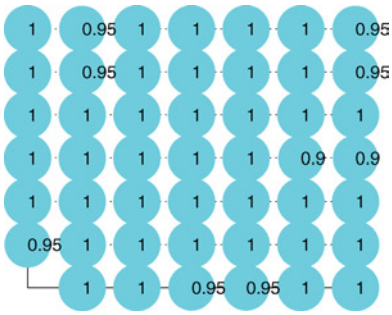


Fig. 13 Individual delivery ratios of LGR with RBC.

Table 3 Performance of LGR with RBC.

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

the trace is 6.66 messages/second, when the delay of each data message is 0 and all the data messages in the trace are received by the base station.

IX. Advantages and Limitations

The LGR protocol has the following advantages (over other routing protocols in sensor networks): simplicity, load balancing, fast fault recovery, security without cryptography, and supporting soft real-time applications. Next, we discuss each of the advantages of the protocol.

i) Simplicity The LGR protocol is simple and so it consumes a small percentage of the network resources (specially compared to traditional distance-vector routing protocols). In the LGR protocol, each sensor is required to store no more than 10–15 bytes of routing information that includes P, pid, and trc, and to send a connected message that has only one message field every 20 seconds (or less frequently).

ii) Load balancing Each sensor distributes data traffic over all its potential parents, and so the data items from the same source sensor are likely to follow different paths to the base station. 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) 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.

iv) 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 routing 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 LGR protocol, each sensor has a set of potential parents in the routing 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 LGR protocol provides this security feature without encrypting and decrypting connected messages.

v) Supporting soft real-time applications The LGR protocol inherently bounds the number of hops that need to be traversed by a data message from any sensor to the base station. Thus, the delivery delay from any sensor to the base station can be predicted accurately in many soft real-time applications, as discussed in.[22] The number of hops C from a sensor (i, j) that has a parent to the base station is bounded as follows.

$$C \leq \max\left(\left\lceil \frac{i}{H} \right\rceil + j, i + \left\lceil \frac{j}{H} \right\rceil\right)$$

On the other hand, the LGR protocol has the following limitations: initial setup requirement, limited communication patterns, limited connectivity, and no support for mobility. Next, we discuss each limitation of the protocol.

i) Initial setup requirement The LGR protocol needs to be set up. Each sensor may need to use GPS or a localization service[15,16] to compute a distinct identifier in a logical grid, specially for a large network. Also we perform off-line experiments to estimate link quality so that we use these results to identify reliable links. However, once this setup is over, the sensors can take all the advantages that the LGR protocol provides.

ii) Limited communication patterns The LGR 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 LGR 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 find a path to the base station, even when there is a possible path to the base station. However, the probability that this case happens in the network is very small. We showed in Section VI that even when 50% of sensors in the network fail-stop, 84% of the 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.

X. Related Work

Several experimental studies have been done to understand the nature of dynamic and lossy wireless sensor networks. Ganesan et al.[8] performed an experimental study on simple flooding, and reported the complex behavior of flooding, such as non-uniform flood propagation, long links, and asymmetric links. Zhao and Govindan[5] investigated packet delivery performance in different indoor and outdoor environments. They showed that a gray area where reception rate varies dynamically exists over relatively long links, and that the probability of packet loss changes over different traffic patterns. Similar study was performed in.[9] They observed that there is no clear correlation between packet delivery and distance over relatively long links. The link quality estimated by periodic beacon exchanges usually reflects the link quality in the absence of bursty data traffic in sensor networks. Zhang et al.[11] showed that a link quality changes significantly when the traffic pattern within the network changes. Thus, it is hard to accurately estimate link quality, especially under bursty traffic since different bursts of traffic usually occur sparsely, and each burst may finish within a short period.

Woo et al.[6] investigated link quality estimation and neighborhood management in dynamic and lossy sensor networks. They developed a routing protocol, where each sensor selects a path with the minimal number of retransmissions based on the estimated link quality. Seada et al.[7] proposed link selection strategies based on estimated link quality for geographic routing protocols. Under bursty traffic, the estimated link quality may not be accurate and stable, resulting in low message delivery rate.

In geographic routing protocols surveyed in,[23] physical location information is used to route data messages. Each node maintains its one-hop neighborhood information, and forwards a message to one of its neighbors that is closer to the destination. As discussed in,[7] these protocols can select unreliable long links to forward a message to the destination. In the LGR protocol, each sensor computes a small number of its potential parents using its identifier, and maintains its current parent in the routing tree. Also it can inherently bound the number of hops needed to make by a message to reach the base station.

He et al.[24] proposed a real-time routing protocol that utilizes location information. To support end-to-end soft real-time communication, each sensor dynamically estimates single hop delay with all its neighbors, and uses the estimated delay to select the next hop that satisfies a desired speed. Thus, the delivery delay is proportional to the distance between the source and the destination. The major focus of the LGR protocol is to provide reliable delivery (of bursty data traffic) to the base station, yet our protocol supports soft real-time applications using the logical grid structure. The experimental results showed that the LGR protocol has successfully delivered data messages with acceptable delay for a tracking application in.[1,12]

Several protocols were proposed to form a virtual grid using location information and utilize the virtual grid for routing, data dissemination, etc. In GRID,[25] the grid is used to limit flooding in mobile ad hoc networks for route discovery, packet relay, and route maintenance. In TTDD,[26] each source proactively builds a grid structure and only sensors in grid points maintain the forwarding information of the source. Thus, a mobile sink can flood queries within a local grid cell and receive data message continuously. TIGR[14] presented an algorithm to compute the transmission and reception schedule for each grid point. Based on the schedule, a sensor can route messages without message collision, and can go to sleep to save energy. In the LGR protocol, a sensor can go to sleep, if each of its potential children has at least one awake potential parent in the logical grid.

Routing protocols were proposed to build virtual coordinates without location information and route messages based on virtual coordinates.[22,27,28] The computation of virtual coordinates is usually based on hop counts from anchors. In,[10] sensors in a network form a layered multi-hop structure based on hop counts to the base station, and each sensor forwards messages to one of its neighbors that have a smaller hop count than itself. These protocols build a routing tree based on hop counts to certain sensors. Therefore, they take advantage of all long links in sensor networks and they may use unreliable long links.

In Directed Diffusion,[29] a sink periodically floods an interest represented by attributes and their values, rather than explicit addresses. Each sensor caches received interests and maintains several gradients for a received interest, up to one per neighbor. Thus, when a sensor has some event, the sensor sends the event to each neighbor that has a gradient for the matching interest, establishing multiple paths between the sink and the source. One of the paths with low delay delivery will be reinforced. The LGR protocol focuses on the reliable message delivery to the fixed base station with small memory requirement, for event-based sensor network applications such as tracking.

A self-stabilizing algorithm was proposed to maintain a spanning tree in a network.[30] This algorithm requires a pre-specified bound on the network size, and it is not guaranteed to stabilize within a time period proportional to the diameter of the network. On the other hand, the LGR protocol stabilizes, without any prior knowledge of the network size or diameter, within a time period proportional to the diameter of the network.

A loop-free distance-vector routing algorithm was proposed in.[31] This algorithm uses diffusing computations (which grow by sending queries and shrink by receiving replies) to maintain distances and avoid forming loops. Note that the LGR protocol needs one-way propagation from the root which is the base station to leaves in the routing tree.

The algorithms presented in[32–35] were designed to localize the impact of faults in routing. The LGR protocol does not consider fault containment, but the mechanisms used by these papers can be applied to the LGR protocol. Detailed study of this is beyond the scope of this paper.

XI. Concluding Remarks

In this paper, we presented the LGR protocol that maintains an incoming spanning tree rooted at the base station to route data messages from any sensor to the base station. In Section IV, we described a basic version of the protocol where each sensor chooses one of its potential parents to be its parent in the routing tree. In Section VI, we described an extended version of the protocol where each sensor can have a foster parent in the routing tree if all its potential parents fail-stop. The experimental results in Section VIII showed that the protocol delivers 72–99% of data messages to the base station under bursty and heavy traffic.

The basic version of the LGR protocol is loop-free, since each sensor chooses only one of its potential parents to be its parent in the routing tree. Thus, starting from any configuration, the protocol converges to a legitimate configuration, without forming any loops. In the extended version of the LGR protocol, temporary loops may be formed. This is because a sensor u can choose any sensor from which sensor u receives a connected message to be its foster parent in the routing tree, when its all potential parents fail-stop. Although this protocol is not loop-free, the convergence time of this protocol is still proportional to the diameter of the sensor network as we showed in Section VII.

The LGR protocol can be easily modified to support multiple base stations. In,[12] the protocol was implemented such that each sensor has 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.

The LGR protocol does not require that all sensors in a network be always awake to maintain a routing tree whose root is the base station. A sensor in the network can be connected to the routing tree if the sensor has at least one awake potential parent (or one awake potential foster parent). Energy saving techniques, such as,[36,37] can be easily incorporated with the LGR protocol to allow some sensors in the network to sleep dynamically.

Acknowledgment

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Contract F33615-01-C-1901. We are grateful to anonymous referees for their helpful comments.

References

[1]Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., and Miyashita, M., "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Computer Networks (Elsevier)*, Vol. 46, No. 5, December 2004, pp. 605–634.

[2]Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E., "Wireless Sensor Networks: A Survey," *Computer Networks, Elsevier Science*, Vol. 38, No. 4, 2002, pp. 393–422.

[3]Mainwaring, A., Polastre, J., Culler, D., and Anderson, J., "Wireless Sensor Networks for Habitat Monitoring," *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[4]Choi, Y., *Design and Analysis of Self-Stabilizing Sensor Protocols*, Ph.D. thesis, University of Texas at Austin, in progress.

[5]Zhao, J. and Govindan, R., "Understanding Packet Delivery Performance In Dense Wireless Sensor Networks," *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, November 2003.

[6]Woo, A., Tony, T., and Culler, D., "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, 2003.

[7]Seada, K., Zuniga, M., Helmy, A., and Krishnamachari, B., "Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks," *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[8]Ganesan, D., Krishnamurthy, B., Woo, A., Culler, D., Estrin, D., and Wicker, S., "An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks," *IRP-TR-02-003*, 2002.

[9]Cerpa, A., Busek, N., and Estrin, D., "SCALE: A tool for Simple Connectivity Assessment in Lossy Environments," *CENS Technical Report 21*, September 2003.

[10]Ding, J., Sivalingam, K. M., Kashyapa, R., and Chuan, L., "A Multi-Layered Architecture and Protocols for Large-Scale Wireless Sensor Networks," *Proceedings of IEEE Semiannual Vehicular Technology Conference Fall*, 2003.

[11]Zhang, H., Arora, A., and Sinha, P., "Learn on the Fly: Quiescent Routing in Wireless Sensor Networks," *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, 2006.

[12]Arora, A., Ramnath, R., Ertin, E., Sinha, P., Bapat, S., Naik, V., Kulathumani, V., Zhang, H., Cao, H., Sridhara, M., Kumar, S., Seddon, N., Anderson, C., Herman, T., Trivedi, N., Zhang, C., Gouda, M., Choi, Y., Nesterenko, M., Shah, R., Kulkarni, S., Aramugam, M., Wang, L., Culler, D., Dutta, P., Sharp, C., Tolle, G., Grimmer, M., Ferriera, B., and Parker, K., "ExScal: Elements of an Extreme Scale Wireless Sensor Network," *Proceedings of 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005)*, 2005.

[13]Xu, Y., Heidemann, J., and Estrin, D., "Geography-informed Energy Conservation for Ad-hoc Routing," *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.

[14]Friedman, R. and Korland, G., "Timed Grid Routing (TIGR) Bites off Energy," *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005)*, 2005.

[15]Hightower, J. and Borriello, G., "Location Systems for Ubiquitous Computing," *IEEE Computer*, Vol. 34, No. 8, August 2001, pp. 57–66.

[16]Simon, G., Maroti, M., Ledeczi, A., Balogh, G., Kusy, B., Nadas, A., Pap, G., Sallai, J., and Frampton, K., "Sensor Network-Based Countersniper System," *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, 2004, pp. 1–12.

[17]Gouda, M., Choi, Y., and Arora, A., "Sentries and Sleepers in Sensor Networks," *Proceedings of 8th International Conference on Principles of Distributed Systems (OPODIS 2004)*, December 2004.

[18]Arora, A. and Gouda, M., "Closure and Convergence: A Foundation of Fault-Tolerant Computing," *IEEE Transactions on Software Engineering*, Vol. 19, No. 11, 1993, pp. 1015–1027.

[19]Berkeley DARPA-NEST team, "Wireless Embedded Systems," *http://webs.cs.berkeley.edu/*.

[20]"Tinyos," http://www.tinyos.net.

[21]Zhang, H., Arora, A., Choi, Y., and Gouda, M., "Reliable Bursty Convergecast in Wireless Sensor Networks," *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2005.

[22]Cao, Q. and Abdelzaher, T., "A Scalable Logical Coordinates Framework for Routing in Wireless Sensor Networks," *Proceedings of the IEEE Real-time Systems Symposium (IEEE RTSS'04)*, Lisbon, Portugal, December 2004.

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

[24]He, T., Stankovic, J., Lu, C., and Abdelzaher, T., "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *Proceedings of the International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[25]Liao, W.-H., Tseng, Y.-C., and Sheu, J.-P., "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks," *Telecommunication Systems*, Vol. 18, No. 1, 2001, pp. 37–60.

[26]Ye, F., Luo, H., Cheng, J., Lu, S., and Zhang, L., "A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *Proceedings of the ACM International Conference on Mobile Computing and Networking*, 2002.

[27]Newsome, J. and Song, D., "GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information," *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, 2003.

[28]Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., and Stoica, I., "Geographic routing without location information," *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom 2003)*, 2003.

[29]Intanagonwiwat, C., Govindan, R., and Estrin, D., "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks*, 2000.

[30]Arora, A. and Gouda, M., "Distributed Reset," *IEEE Transactions on Computers*, Vol. 43, No. 9, 1994, pp. 1026–1038.

[31]Garcia-Lunes-Aceves, J. J., "Loop-Free Routing Using Diffusing Computations," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 1, 1993, pp. 130–141.

[32]Kutten, S. and Patt-Shamir, B., "Time-adaptive Self Stabilization," *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing (PODC 1997)*, August 1997, pp. 149–158.

[33]Ghosh, S., Gupta, A., Herman, T., and Pemmaraju, S. V., "Fault-containing self-stabilizing algorithms," *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing (PODC 1996)*, 1996, pp. 45–54.

[34]Beauquier, J. and Herault, T., "Fault-local stabilization: the shortest path tree," *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS 2002)*, 2002.

[35]Arora, A. and Zhang, H., "LSRP: Locat Stabilization in Shortest Path Routing," *IEEE-IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2003, pp. 139–148.

[36]Cerpa, A. and Estrin, D., "ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies," *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, June 23–27 2002.

[37]Hohlt, B., Doherty, L., and Brewer, E., "Flexible Power Scheduling for Sensor Networks," *IEEE and ACM Third International Symposium on Information Processing in Sensor Networks*, April 2004.

Shlomi Dolev
Associate Editor