Stabilization of Flood Sequencing Protocols in Sensor Networks

Young-ri Choi, Chin-Tser Huang, *Member*, *IEEE*, and Mohamed G. Gouda, *Member*, *IEEE*

Abstract—Flood is a communication primitive that can be used by the base station of a sensor network to send a copy of a message to every sensor in the network. When a sensor receives a flood message, the sensor needs to check whether it has received this message for the first time and so this message is fresh, or it has received the same message earlier and so the message is redundant. In this paper, we discuss a family of four flood sequencing protocols that use sequence numbers to distinguish between fresh and redundant flood messages. These four protocols are: a sequencing free protocol, a linear sequencing protocol, a circular sequencing protocol, and a differentiated sequencing protocol. We analyze the self-stabilization properties of these four flood sequencing protocols. We also compare the performance of these flood sequencing protocols, using simulation, over various settings of sensor networks. We conclude that the differentiated sequencing protocol has better stabilization property and provides better performance than those of the other three protocols.

Index Terms—Self-stabilization, flood sequencing protocol, sequence numbers, sensor networks.

1 Introduction

Flood is a communication primitive that can be used by the base station of a sensor network to send a copy of a message to every sensor in the network. The execution of a flood starts by the base station sending a message to all its neighbors. When a sensor receives a message, the sensor needs to check whether it has received this message for the first time or not. Only if the sensor has received the message for the first time, the sensor keeps a copy of the message and may forward the message to all its neighbors. Otherwise, the sensor discards the message.

To distinguish between "fresh" flood messages that a sensor should keep and "redundant" flood messages that a sensor should discard, the base station selects a sequence number and attaches it to a flood message before the base station broadcasts the message. When a sensor receives a flood message, the sensor determines based on the sequence number in the received message if the message is fresh or redundant. The sensor accepts the message if it is fresh and discards the message if it is redundant. We call a protocol that uses sequence numbers to distinguish between fresh and redundant flood messages a *flood sequencing protocol*.

In a flood sequencing protocol, when a fault corrupts the sequence numbers stored in some sensors in a sensor network, the network can become in an illegitimate state where the sensors discard fresh flood messages and accept redundant flood messages. Therefore, a flood sequencing protocol should be designed such that if the protocol ever reaches an illegitimate state due to some fault, the protocol is guaranteed to converge back to its legitimate states where every sensor accepts every fresh flood message and discards every redundant flood message.

In this paper, we discuss a family of four flood sequencing protocols. These four protocols are: a *sequencing free* protocol, a *linear sequencing* protocol, a *circular sequencing* protocol, and a *differentiated sequencing* protocol. We analyze the stabilization properties of these four protocols. For each of the protocols, we first compute an upper bound on the convergence time of the protocol from an illegitimate state to legitimate states. Second, we compute an upper bound on the number of fresh flood messages that can be discarded by each sensor during the convergence. Third, we compute an upper bound on the number of redundant flood messages that can be accepted by each sensor during the convergence. We also compare the performance of these protocols, using simulation, over various settings of sensor networks.

The rest of the paper is organized as follows: In Section 2, we discuss related work and motivation of the flood sequencing protocols. In Section 3, we present a model of the execution of a sensor network. In Section 4, we give an overview of a flood sequencing protocol. We present the four flood sequencing protocols and analyze their stabilization properties in Sections 5-8. In Section 9, we show the simulation results of these protocols, and in Section 10, we discuss how frequent floods affect the performance of the differentiated sequencing protocol, and how the protocol can be extended for multiple sources. We finally make concluding remarks in Section 11.

2 Related Work and Motivation

The practice of using sequence numbers to distinguish between fresh and redundant flood messages has been

- *Y.-r. Choi is with Korea Institute of Science and Technology Information, 335 Gwahangno, Yuseong-gu, Daejeon, 305-806, Republic of Korea. E-mail: ychoi@kisti.re.kr.*
- *C.-T. Huang is with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St, Swearingen Center, Room#3A59, Columbia, SC 29208. E-mail: huangct@cse.sc.edu.*
- *M.G. Gouda is with the Department of Computer Sciences, University of Texas at Austin, 1 University Station (C0500), Austin, TX 78712-0233. E-mail: gouda@cs.utexas.edu.*

Manuscript received 5 Oct. 2008; revised 1 June 2009; accepted 23 July 2009; published online 30 July 2009.
Recommended for acceptance by S. Das.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-10-0403.
Digital Object Identifier no. 10.1109/TPDS.2009.126.

adopted by most flood protocols in the literature. In other words, most flood protocols "employ" some flood sequencing protocols to distinguish between fresh and redundant flood messages. A flood sequencing protocol can be designed in various ways, depending on several design decisions such as how the next sequence number is selected by the base station, how each sensor determines based on the sequence number in a received message if the received message is fresh or redundant, and what information the base station and each sensor stores in its local memory. Unfortunately, flood sequencing protocols have been used without full investigation of their design decisions.

There have been earlier efforts to study flood protocols in ad hoc networks [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], and in sensor networks [13], [14], [15], [16]. It is important to state that these protocols focus on reducing the total number of retransmissions or forwarding (sensor) nodes for a flood message, while the flood sequencing protocols focus on distinguishing between fresh and redundant messages, to prevent nodes from forwarding the same message more than once. Thus, these protocols compute if a node can reach additional nodes by forwarding a received message, and make only those nodes that can reach additional nodes forward the message, based on probability [2], [3], location information [2], [6], [9], [13], [14], [16], or neighbor and history information [4], [5], [10], [11], [12].

The flood protocols discussed in [3], [4], [5], [6], [13], [15] assume (explicitly or implicitly) that when a node receives a flood message, the node can figure out whether it has received this message for the first time or not, without specifying any mechanism to achieve this.

In [2], [16], it was suggested to associate a sequence number with each flood message. The flood protocols discussed in [7], [8], [14] propose to attach a unique identifier or sequence number to each flood message and make each node maintain a list of identifiers that it has received recently. Similarly, it was suggested in [9] that each node maintains a list of flood messages received by the node recently. In [11], for a recently received flood message, each node maintains an entry of the source address, sequence number, and lifetime. However, in these protocols, any details on how sequence numbers or identifiers are used by nodes, how many identifiers or messages each node maintains, when a node deletes an identifier or a message from the list, or how the lifetime of a message is determined (i.e., the design decisions of their flood sequencing protocols) were not specified.

In Scalable Reliable Multicast (SRM) [17], when a receiver in a multicast group detects that it has a missing data message, it attempts to retrieve the message from any node in the group by requesting retransmission. This work is based on the assumption that each data message has a unique and persistent name, and it utilizes application data units to name messages (such as sector 5 of a file "paper.pdf"). In a flood sequencing protocol, sensors can use sequence numbers in a limited range for flood messages. Thus, the sensors cannot identify a message uniquely based on the sequence number of the message, and cannot use the sequence number for requesting retransmission and replying to a request. The protocols in [18], [19] use named data that is specific to applications for dissemination and routing in sensor networks. However, a flood sequencing protocol can be used, before any application is deployed in the network. Thus, using named data is not suitable for a flood sequencing protocol.

A flood sequencing protocol is important, since the fault tolerance property of a sensor network is affected by a flood sequencing protocol used in the network. When a fault corrupts the sequence number stored in some sensor in the network, the sensor may discard fresh flood messages and accept redundant flood messages. The number of fresh flood messages discarded by the sensor and the number of redundant flood messages accepted by the sensor, before the network reaches a legitimate state, are different depending on which flood sequencing protocol is used in the network. Therefore, we need to study various flood sequencing protocols and analyze the stabilization properties of these protocols. The stabilization properties of the flood sequencing protocols are useful for sensor network designers or developers to select a proper flood sequencing protocol that satisfies the needs of a target sensor network.

In practice, a flood sequencing protocol is used with a flood protocol that may use other techniques to improve the performance of flood such as reliability or efficiency (as discussed above). In this paper, each of the flood sequencing protocols is described focusing on how sequence numbers are used by sensors, and it is not described as a specific flood protocol. Note that the stabilization property of a flood protocol is affected by that of a flood sequencing protocol used in the flood protocol. If the flood protocol does not maintain any extra state such that it is based on probability [3], [16], the stabilization property of the flood protocol is the same as that of the used flood sequencing protocol. If the flood protocol maintains extra state such that it is based on neighbor information [4], [10], the stabilization property of the flood protocol also depends on how the extra state in each sensor is stabilized.

3 Model of Sensor Networks

In this section, we describe a formal model of the execution of a sensor network, which was introduced first in [20]. This model accommodates several characteristics of sensor networks such as unavoidable local broadcast, probabilistic message transmission, asymmetric communication, and message collision. We use the model to specify our flood sequencing protocols, verify the stabilization properties of these protocols, and develop our simulation of these protocols.

Topology of sensor networks. The *topology* of a sensor network is a directed graph where each node represents a distinct sensor in the network and where each directed edge is labeled with some probability. A directed edge (u, v), from a sensor u to a sensor v, that is labeled with probability p (where $p > 0$) indicates that if sensor u sends a message, then this message arrives at sensor v with probability p (provided that neither sensor v nor any "neighboring sensor" of v sends another message at the same time). In this work, two values 0.95 and 0.5 are selected for p. We will discuss some experiments on sensors that led us to this choice of values in Section 9. If the topology of a sensor network has a directed edge from a sensor u to a sensor v, then u is called an *in-neighbor* of v and v is called an *out-neighbor* of u.

Sensor network execution. We assume that during the execution of a sensor network, the real time passes through discrete instants: instant 1, instant 2, instant 3, and so on. The time periods between consecutive instants are equal. The different activities that constitute the execution of a sensor network occur only at the time instants, and not in the time periods between the instants. We refer to the time period between two consecutive instants t and $t+1$ as a *time unit* $(t,\ t+1)$.

A sensor is specified as a program that has global constants, local variables, one time-out action, and one receiving action.

At a time instant t, if the time-out of a sensor u expires, then u executes its time-out action at t. Executing the time-out action of sensor u at t causes u to update its local variables, and to send at most one message at t. It also causes u to execute the statement "timeout-after <expression>" which causes the time-out of u to expire (again) after k time units, where k is the value of <expression> at the time unit $(t,\ t+1)$. The time-out 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>>
```

To keep track of its time-out, each sensor u has an implicit variable named "timer.u." In each time unit between two consecutive instants, timer.u has a fixed positive integer value. If the value of timer.u is k, where $k > 1$, in a time unit $(t-1,\ t)$, then the value of timer.u is $k-1$ in the time unit $(t,\ t+1)$. On the other hand, if the value of timer.u is 1 in a time unit $(t-1,\ t)$, then sensor u executes its time-out action at instant t. Moreover, since sensor u executes the statement "timeout-after <expression>" as part of executing its time-out action, the value of timer.u in the time unit $(t,\ t+1)$ is the value of <expression> in the same time unit.

If a sensor u executes its time-out action and sends a message at an instant t, then an out-neighbor v of u receives a copy of the message at the same time instant t with probability p, where p is the label of edge (u, v) in the topology, provided that the message sent by u does not *collide* with another message sent by v or another in-neighbor of v at t. If the message sent by u collides with another message, then v receives no message at t. We will discuss details of probabilistic message transmission and message collision in Section 9. Sending operations and their corresponding receiving operations are executed synchronously in the network, and a sensor cannot send and receive messages at the same time instant.

If a sensor u receives a message at instant t, then u executes its receiving action at t. 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>." The receiving action of sensor u is of the following form:

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

A *state* of a sensor network protocol is defined by a value for each variable and timer.u for each sensor u in the protocol. We use the notation <var>.u to denote the value of variable <var> at sensor u. (Note that a state of the protocol corresponds to a time unit.)

```
1:  sensor 0                          {base station}
2:  const hmax  : integer,            {max hop count}
3:        f     : integer             {flood period}
4:  var   slast : integer             {last seq number}
5:  begin
6:      timeout-expires →   {generate new msg}
7:                          {select a sequence number slast}
8:                          send data(hmax,slast);
9:                          timeout-after f
10: end
```

Fig. 1. A specification of sensor 0 in a flood sequencing protocol.

During the execution of a sensor network protocol, several faults can occur, resulting in corrupting the state of the protocol arbitrarily. Examples of these faults are wrong initialization, memory corruption, message corruption, and sensor failure and recovery. We assume that these faults do not continuously occur in the network.

4 Overview of a Flood Sequencing Protocol

In this section, we give an overview of a flood sequencing protocol as well as a flood protocol that is used with the flood sequencing protocol.

Consider a network that has n sensors. In this network, sensor 0 is the base station and can initiate message floods over the network. To initiate the flood of a message, sensor 0 selects a sequence number $slast$ for the message, and sends the message of the form data($hmax$, $slast$), where $hmax$ is the maximum number of hops to be made by this data message in the network.

If sensor 0 initiates one flood and shortly after initiates another flood, some forwarded messages from these two floods can collide with one another causing many sensors in the network not to receive the message of either flood, or (even worse) not to receive the messages of both floods.

To prevent message collision across consecutive flood messages, once sensor 0 broadcasts a message, it needs to wait enough time until this message is no longer forwarded in the network, before broadcasting the next message. The time period that sensor 0 needs to wait after broadcasting a message and before broadcasting the next message is called the *flood period*. The flood period consists of f time units. (A lower bound on the value of f is computed below.) Thus, after sensor 0 broadcasts a message, it sets its time-out to expire after f time units in order to broadcast the next message. A formal specification of sensor 0 in a flood sequencing protocol is given in Fig. 1. Note that sensor 0 does not receive any messages.

Each sensor u that is not sensor 0 keeps track of the last sequence number accepted by u in a variable called $slast$. When sensor u receives a data(h, s) message, the sensor decides whether it accepts the message based on the values of $slast$ and s, and forwards it as a data$(h-1,\ s)$ message, provided $h > 1$.

To reduce the probability of message collision, any sensor u, that decides to forward a message, chooses a random period whose length is chosen uniformly from the range $1..tmax$, and sets its time-out to expire after the

```
1:  sensor u:1 .. n − 1
2:  const  hmax   : integer,        {max hop count}
3:         tmax   : integer         {max forwarding period}
4:  var    h,hlast : 1 .. hmax,     {rcvd,last hop count}
5:         s, slast : integer,      {rcvd,last seq number}
6:         new    : boolean         {true if u has msg to forward}
7:  begin
8:    timeout-expires → if new →     new := false;
9:                                   send data(hlast, slast)
10:                     [] ¬ new →   skip
11:                     fi; timeout-after random(1,tmax)

12:   [] rcv data(h, s) → if {this msg is fresh} →
13:                          {accept msg} slast := s;
14:                          if h>1 →  new := true;
15:                                    hlast := h − 1
16:                          [] h≤ 1 → skip
17:                          fi
18:                     [] {this msg is redundant} →
19:                          {discard msg} skip
20:                     fi
21: end
```

Fig. 2. A specification of sensor u in a flood sequencing protocol.

chosen random period, so that u can forward the received message at the end of the random period. This random time period is called the *forwarding period*. A sensor u maintains a variable called new. The value of new is true only when u is in the forwarding period. A formal specification of sensor u is given in Fig. 2. Sensor u also maintains a received data message that u will forward later, even though this is not explicitly specified in the formal specification. Note that in the protocol, the value of timer.0 is at most f time units, and the value of timer.u is at most $tmax$. This is maintained by the executions of the protocol.

This flood sequencing protocol is in a legitimate state iff it satisfies the following two conditions:

1. Every time sensor 0 initiates a new flood, previous flood messages whether initiated by sensor 0 legitimately or other sensors illegitimately due to some fault are no longer forwarded in the network.
2. Every sensor u accepts every fresh flood message, and discards every redundant flood message.

We give some explanation concerning the first condition. If timer.$0 = 1$ in time unit $(t - 1,\ t)$, sensor 0 executes its time-out action at t, and initiates a new flood at t. (Note that after sensor 0 initiates a new flood at t, timer.0 becomes f in $(t,\ t + 1)$.) In $(t - 1,\ t)$, for every sensor u, where $u \neq 0$, u should not have any previous message that has been received but has not been forwarded yet. Thus, if timer.$0 = 1$, for every u, new.u should be false.

Next, we compute the lower bound on the flood period f. The proofs of all the theorems in this paper are presented in the Appendix.

Theorem 0.

$$f \geq (hmax - 1) * tmax + 1.$$

To analyze each of the four flood sequencing protocols, we use the following value for the flood period f.

$$f = hmax * tmax + 1.$$

(We choose this value for f, instead of the minimum value $(hmax - 1) * tmax + 1$, to keep our proofs of the stabilization properties simple.)

Note that the above flood period is computed to guarantee that no two consecutive flood messages ever collide with each other. In a typical execution of the protocol, each sensor chooses its forwarding period at random in the range $1..tmax$, and so most sensors likely receive the flood messages within $(hmax - 1) * tmax/2$ time units. Moreover, the flood of a message is affected by the topology of the network. Therefore, we can use only half or even less of the flood period without significantly degrading the stabilization property and performance of a flood sequencing protocol. In Section 10, we will discuss the effect of frequent floods on the differentiated sequencing protocol.

5 First Protocol: Sequencing Free

In this section, we discuss a first flood sequencing protocol where no sequence number is attached to each flood message. In this protocol, no sensor can distinguish between fresh and redundant flood messages, resulting that the sensor accepts every received message. This protocol is called the *sequencing free* protocol.

To initiate the flood of a new message, sensor 0 sends a data($hmax$) message, and then sets its time-out to expire after f time units to broadcast the next message. The time-out action of sensor 0 is specified as follows:

```
1:  timeout-expires →  {generate new msg}
2:                     send data(hmax);
3:                     timeout-after f
```

When sensor u receives a data(h) message, u always accepts the message. Sensor u forwards the message as data($h - 1$), if $h > 1$ in the received message and $new = false$ in u. The time-out and receiving actions of sensor u are specified as follows:

```
1:  timeout-expires → if new →      new := false;
2:                                  send data(hlast)
3:                    [] ¬ new →    skip
4:                    fi; timeout-after random(1,tmax)

5:  [] rcv data(h) →  {accept msg}
6:                    if h>1 ∧ ¬ new → new := true;
7:                                     hlast := h − 1
8:                    [] h≤ 1 ∨ new → skip
9:                    fi
10: end
```

A state S of the sequencing free protocol is *legitimate* iff either S is a state where the predicate

$$(\text{timer}.0 = 1) \wedge (\text{for all } u, u \neq 0, \text{new}.u = false) \qquad \text{(P1)}$$

holds or S is a state that is reachable from a state, where this predicate holds, by some execution of the protocol.

It follows from this definition that if the protocol is executed starting from a legitimate state, then every time sensor 0 initiates a new flood, previous flood messages are no longer forwarded in the network.

Unlike the other flood sequencing protocols in this paper, each flood message has no associated sequence number. Thus, sensors do not have variable $slast$ and/or

variable s, and a legitimate state is defined as a state that satisfies only the first condition in Section 4.

The stabilization property of the sequencing free protocol can be stated by the following three theorems: Theorem 1A gives an upper bound on the convergence time of the protocol from an illegitimate state to legitimate states. Theorem 1B gives an upper bound on the number of fresh messages that can be discarded by each sensor during the convergence. Theorem 1C gives an upper bound on the number of redundant messages that can be accepted by each sensor during the convergence. (In general, the stabilization property of each of the other three protocols can be stated by three theorems: Theorem iA, Theorem iB, and Theorem iC, where i = 2, 3, and 4.)

Theorem 1A. *In the sequencing free protocol, starting from any illegitimate state, the protocol reaches a legitimate state within $2 * f$ time units, and continues to execute within legitimate states.*

Theorem 1B. *In the sequencing free protocol, starting from any illegitimate state, every sensor discards no fresh message (before the protocol converges to a legitimate state).*

Note that starting from any legitimate state, every sensor discards no fresh message, since the sensor accepts every received message.

Theorem 1C. *In the sequencing free protocol, starting from any illegitimate state, every sensor accepts at most $2 * f$ redundant messages (before the protocol converges to a legitimate state).*

Note that even starting from any legitimate state, the sensor cannot distinguish between fresh and redundant flood messages. The number of redundant copies of the same message accepted by a sensor u depends on the value of $hmax$ and the network topology. In the worst case, u can accept a redundant copy of the same message at each time instant during the flood period of the message. Thus, starting from any legitimate state, every sensor accepts at most f redundant copies of the same message.

6 Second Protocol: Linear Sequencing

In this section, we discuss a second flood sequencing protocol where each flood message carries a unique sequence number that is linearly increased, and so a sensor accepts a flood message that has a sequence number larger than the last sequence number accepted by the sensor. This protocol is called the *linear sequencing* protocol.

Each flood message in this protocol is augmented with a unique sequence number. Whenever sensor 0 broadcasts a new message, sensor 0 increases the sequence number of the last message by one, and attaches the increased sequence number to the message. The time-out action of sensor 0 is given as follows:

```
1:  timeout-expires →  {generate new msg}
2:                     slast := slast + 1;
3:                     send data(hmax,slast);
4:                     timeout-after f
```

When sensor u receives a data(h, s) message, sensor u accepts the message if $s > slast$, and forwards the message if $h > 1$. Otherwise, sensor u discards the message. The receiving action of u is given as follows:

```
1:  [] rcv data(h, s) → if s > slast →  {accept msg} slast := s;
2:                        if h>1 →  new := true;
3:                                  hlast := h − 1
4:                        [] h≤ 1 → skip
5:                        fi
6:                      [] s ≤ slast →  {discard msg} skip
7:                      fi
```

A state S of the linear sequencing protocol is *legitimate* iff either S is a state where the predicate

$$(\text{timer}.0 = 1) \wedge (\text{for all } u, u \neq 0, \text{new}.u = false \wedge \text{slast}.u \leq \text{slast}.0) \quad \text{(P2)}$$

holds or S is a state that is reachable from a state, where this predicate holds, by some execution of the protocol.

It follows from this definition that if the protocol is executed starting from a legitimate state, then every time sensor 0 initiates a new flood, previous flood messages are no longer forwarded in the network, and the new flood message has a sequence number that is larger than every slast.u in the network, so that every u accepts the message.

Let k be the maximum value between 1 and k', where k' is the maximum difference $slast.u - slast.0$ for any sensor u in the network at an initial state. Note that the value of k is finite but it is unbounded.

Theorem 2A. *In the linear sequencing protocol, starting from any illegitimate state, the protocol reaches a legitimate state within $(k + 1) * f$ time units, and continues to execute within legitimate states.*

Theorem 2B. *In the linear sequencing protocol, starting from any illegitimate state, every sensor discards at most $(k + 1) * f$ fresh messages (before the protocol converges to a legitimate state).*

Theorem 2C. *In the linear sequencing protocol, starting from any illegitimate state, every sensor accepts at most $n - 1$ redundant messages (before the protocol converges to a legitimate state).*

The linear sequencing protocol requires sensors to use unbounded sequence numbers. Thus, this protocol is very expensive to implement for sensor networks that have limited resources. However, once the protocol starts its execution from any legitimate state, every sensor accepts every fresh message and discards every redundant message under any degree of message loss.

7 Third Protocol: Circular Sequencing

In this section, we discuss a third flood sequencing protocol where each flood message carries a sequence number that is circularly increased within a limited range, and so a sensor accepts a flood message that has a sequence number "logically" larger than the last sequence number accepted by the sensor. This protocol is called the *circular sequencing* protocol.

Each flood message is augmented with a sequence number that has a value in the range 0 .. $smax$, where $smax > 1$. We assume that $smax$ is an even number (to keep our presentation simple).

Whenever sensor 0 broadcasts a new message, sensor 0 increases the sequence number of the last message by one

circularly within the range 0 .. $smax$, and attaches the increased sequence number to the message. The time-out action of sensor 0 is modified as follows:

```
1:  timeout-expires →   {generate new msg}
2:                      slast := (slast + 1) mod (smax+1);
3:                      send data(hmax,slast);
4:                      timeout-after f
```

From the viewpoint of each sequence number s in the range 0 .. $smax$, the range can be divided into two subranges, where one subrange consists of the sequence numbers that are logically "smaller" than s, and the other subrange consists of the sequence numbers that are logically "larger" than s. Thus, sequence number s has $\frac{smax}{2}$ numbers logically smaller than it and $\frac{smax}{2}$ numbers logically larger than it. For example, if $smax = 8$, number 0 is logically smaller than 1, 2, 3, and 4, and is logically larger than 5, 6, 7, and 8.

When a sensor u receives a data(h, s) message, sensor u checks if s is logically larger than $slast$. Sensor u calls the function "Larger$(s, slast)$" that returns true if s is logically larger than $slast$, and otherwise returns false. Sensor u accepts the message if Larger$(s, slast)$ returns true, and forwards it if $h > 1$. The receiving action of sensor u is modified as follows:

```
1:  [] rcv data(h, s) →
2:      if Larger(s,slast) →    {accept msg} slast := s;
3:                              if h>1 →   new := true;
4:                                         hlast := h − 1
5:                              [] h≤ 1 →  skip
6:                              fi
7:      [] ¬Larger(s,slast) →   {discard msg} skip
8:      fi
```

To prove the stabilization property of the circular sequencing protocol, we make an assumption of bounded message loss as follows:

- *Bounded message loss*: Starting from any state, if sensor 0 broadcasts $\frac{smax}{2}$ consecutive flood messages, then every sensor in the network receives at least one of those flood messages.

Two explanations concerning the above assumption are in order. First, the protocol cannot be self-stabilizing without any bound on message loss. For example, consider a scenario where $smax = 8$. Assume that sensor 0 sends a flood message with sequence number 0 and a sensor u accepts the message. If sensor u does not receive the next four (i.e., $\frac{smax}{2}$) consecutive messages with sequence numbers 1, 2, 3, and 4, and later receives a fresh message with sequence number 5, it discards the message since sequence number 5 is not logically larger than sequence number 0. Sensor u also discards the next flood messages with sequence numbers 6, 7, 8, and 0, if it receives them. In this scenario, if sensor u does not receive the flood messages with sequence numbers 1, 2, 3, and 4, it keeps discarding fresh flood messages. Thus, some assumption of bounded message loss is necessary for the stabilization property of the protocol.

Second, the above assumption becomes acceptable if the value of $smax$ is reasonably large enough for a given network setting. Selecting an appropriate value for $smax$ depends on the size of the network, the topology of the network, and a flood sequencing protocol used in the network. (In Section 9, we show how different values are selected for $smax$ depending on these factors.)

A state S of the circular sequencing protocol is *legitimate* iff either S is a state where the predicate

$$
\begin{aligned}
&(\text{timer}.0 = 1) \wedge \\
&(\text{for all } u, u \neq 0, \\
&\quad (\text{new}.u = \text{false}) \wedge \\
&\quad (\text{slast}.u = \text{slast}.0 \vee \\
&\quad \text{slast}.u = (\text{slast}.0 - 1) \bmod (smax + 1) \vee \\
&\quad \cdots \\
&\quad \text{slast}.u = (\text{slast}.0 - \frac{smax}{2} + 1) \bmod (smax + 1) \\
&\quad) \\
&) \wedge \\
&(\text{sensor 0 has already initiated at least } \frac{smax}{2} + 2 \text{ floods})
\end{aligned}
\tag{P3}
$$

holds or S is a state that is reachable from a state, where this predicate holds, by some execution of the protocol.

It follows from this definition that if the protocol is executed starting from a legitimate state, then every time sensor 0 initiates a new flood, previous flood messages are no longer forwarded in the network, and the new flood message has a sequence number that is logically larger than every slast.u in the network, so that every u accepts the message.

Theorem 3A. *In the circular sequencing protocol, starting from any illegitimate state, the protocol reaches a legitimate state within $(smax + 2) * f$ time units, and continues to execute within legitimate states.*

Theorem 3B. *In the circular sequencing protocol, starting from any illegitimate state, every sensor discards at most $(smax + 2) * f$ fresh messages (before the protocol converges to a legitimate state).*

Theorem 3C. *In the circular sequencing protocol, starting from any illegitimate state, every sensor accepts at most $f + 1$ redundant messages (before the protocol converges to a legitimate state).*

Note that starting from any legitimate state, every sensor accepts every fresh message and discards every redundant message under the assumption of bounded message loss.

8 Fourth Protocol: Differentiated Sequencing

In this section, we discuss the last flood sequencing protocol where the sequence numbers of flood messages are in a limited range, similar to the circular sequencing protocol. However, in this protocol, a sensor accepts a flood message if the sequence number of the message is different from the last sequence number accepted by the sensor. This protocol is called the *differentiated sequencing* protocol.

Each flood message is augmented with a sequence number that has a value in the range 0 .. $smax$, where

TABLE 1
Stabilization Properties of the Flood Sequencing Protocols

	Convergence time (time units)	Max # of fresh msgs discarded by u until convergence	Max # of redundant msg accepted by u until convergence	Stabilization property
free	$2 * f$	0	$2 * f$	good
lin	unbounded	unbounded	$n-1$	bad
cir	$(smax+2) * f$	$(smax+2) * f$	$f+1$	good
dif	$(\frac{smax}{2}+2) * f$	$(\frac{smax}{2}+2) * f$	$f+1$	good

TABLE 2
Stable Properties of the Flood Sequencing Protocols

	Max # of fresh msgs discarded by u after convergence	Max # of redundant copies of the same msg accepted by u after convergence	Stable property
free	0	f	bad
lin	0	0	good
cir	0	0	good
dif	0	0	good

$smax > 0$. We assume that $smax$ is an even number (to keep our presentation simple).

Sensor 0 in this protocol is identical to the one in the circular sequencing protocol. However, when a sensor u receives a data(h, s) message, sensor u accepts the message if s is different from $slast$, and forwards the message if $h > 1$. The receiving action of sensor u is modified as follows:

```
1: [] rcv data(h, s) → if s ≠ slast →  {accept msg} slast := s;
2:                      if h>1 →       new := true;
3:                                     hlast := h − 1
4:                      [] h≤ 1 →      skip
5:                      fi
6:                   [] s = slast →    {discard msg} skip
7:                   fi
```

Similar to the circular sequencing protocol, if a sensor does not receive a large number of consecutive flood messages, the differentiated sequencing protocol cannot be self-stabilizing. Thus, the proofs of the stabilization properties of this protocol are based on the assumption of bounded message loss described in Section 7.

A state S of the differentiated sequencing protocol is *legitimate* iff either S is a state where the predicate

$$
\begin{aligned}
&(\text{timer}.0 = 1) \wedge \\
&(\text{for all } u, u \neq 0, \\
&\quad (\text{new}.u = \text{false}) \wedge \\
&\quad (\text{slast}.u = \text{slast}.0 \vee \\
&\quad \text{slast}.u = (\text{slast}.0 - 1) \bmod (smax+1) \vee \\
&\quad\quad \cdots \\
&\quad\quad \text{slast}.u = (\text{slast}.0 - \frac{smax}{2} + 1) \bmod (smax+1) \\
&\quad\quad) \\
&)
\end{aligned}
\tag{P4}
$$

holds or S is a state that is reachable from a state, where this predicate holds, by some execution of the protocol.

It follows from this definition that if the protocol is executed starting from a legitimate state, then every time sensor 0 initiates a new flood, previous flood messages are no longer forwarded in the network, and the new flood message has a sequence number that is different from every slast.u in the network, so that every u accepts the message.

Theorem 4A. *In the differentiated sequencing protocol, starting from any illegitimate state, the protocol reaches a legitimate state within $(\frac{smax}{2}+2) * f$ time units, and continues to execute within legitimate states.*

Theorem 4B. *In the differentiated sequencing protocol, starting from any illegitimate state, every sensor discards at most $(\frac{smax}{2}+2) * f$ fresh messages (before the protocol converges to a legitimate state).*

Theorem 4C. *In the differentiated sequencing protocol, starting from any illegitimate state, every sensor accepts at most $f+1$ redundant messages (before the protocol converges to a legitimate state).*

Note that starting from any legitimate state, every sensor accepts every fresh message and discards every redundant message under the assumption of bounded message loss.

We compare the stabilization properties of the four flood sequencing protocols in Table 1. We also compare the properties of the flood sequencing protocols after convergence (or starting from a legitimate state) in Table 2. We call these properties the *stable properties* of the protocols. In Tables 1 and 2, "free," "lin," "cir," and "dif" represent the sequencing free, linear sequencing, circular sequencing, and differentiated sequencing protocols, respectively. We conclude that the differentiated sequencing protocol has better stabilization and stable properties than those of the other three protocols.

9 Performance Evaluation

9.1 Methodology and Metrics

We have developed a simulator that can simulate the execution of the four flood sequencing protocols, based on our formal model described in Section 3. This simulator monitors the execution of an abstract and generic version of a sensor protocol based on our model. However, our model actually captures unique characteristics of sensor networks such as local broadcast, probabilistic message transmission, and message collision. The simulation based on our model is useful to evaluate the performance of protocol designs, and explore the performance trade-offs between sensor protocols. In our simulation, we explore the implications of design decisions on the flood sequencing protocols, and verify and evaluate our theoretical analysis on the stabilization properties of the protocols, some of which are done under the assumption of bounded message loss. (Performance evaluation based on a specific implementation of the protocols such as TinyOS [21] is a subject of future work.)

In the model, a message can be lost due to probabilistic message transmission and message collision. If a sensor u sends a message at an instant t, then an out-neighbor v of u receives a copy of the message at the same time instant t, provided that the following three conditions hold:

1. A random integer number is uniformly selected in the range 0 .. 99, and this selected number is less than $100 * p$, where p is the probability label of edge (u, v) in the topology.
2. Sensor v does not send any message at instant t. If v sends a message at t, this message collides with the message sent by u (with the net result that v receives no message at t).
3. For each in-neighbor w of v, other than u, if w sends a message at t, then a random integer number is uniformly selected in the range 0 .. 99, and the selected number is at least $100 * p'$, where p' is the probability label of edge (w, v). If the selected number is less than $100 * p'$, then this message sent by w collides with the message sent by u.

Note that sending operations and their corresponding receiving operations are executed at the same instant synchronously. (The value of a time unit is not critical to the current work, but we estimate that the value of the time unit is around 100 milliseconds.)

In this work, two values 0.95 and 0.5 are selected for p, based on some experiments on sensors. In the experiments [22], we measured the percentages of messages received at a sensor v, sent by another sensor u over various distances. The results are summarized in Fig. 3. Similar results are also reported in [23] and [24].

We observe that from Fig. 3 if the distance between u and v is less than 0 .. 38 inches, v receives between 90 and 100 percent of the messages sent by u. If their distance is in the range 38 .. 67 inches, v receives anywhere between 0 and 100 percent of the messages sent by u. If their distance is longer than 67 inches, v receives 0 percent of the messages sent by u. From these observations, we idealize the diagram in Fig. 3 as follows: If their distance is in the range 0 .. x, then the directed edge from u to v can be labeled with probability 0.95. If their distance is in the range x .. y, then the directed edge from u to v can be labeled with probability 0.5. (We refer the reader to [22] for details of the experiments.)

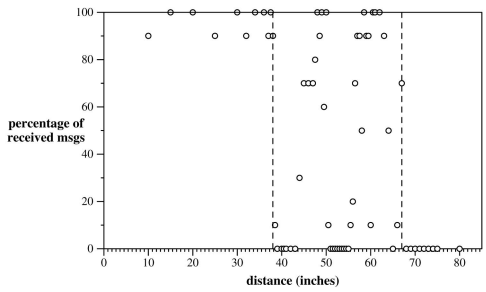

Fig. 3. Percentage of received messages.

In ad hoc and sensor networks, Unit Disk Graph and Quasi Unit Disk Graph models have been used [25], [26], [27]. In both models, a network is represented by an undirected graph, where the euclidean distance between two nodes determines the existence of an edge between them, and an edge between them indicates that they can always communicate with each other directly. Unlike our model, these models do not consider probabilistic message delivery, and asymmetric communication, which are very common in sensor networks.

For the purpose of simulation, a network is an $N * N$ grid where N is the number of sensors in each side of the grid, and the distance between a sensor (i, j) and each of $(i+1,\ j)$, $(i,\ j+1)$, $(i-1,\ j)$, and $(i,\ j-1)$, if it exists, where $0 \leq i, j < N$, is 1. In a grid, sensor 0 is (0,0) which is located at the left-bottom corner. Two values 10 and 20 were used for N, and also the following two types of topologies that have different network density were used.

- A topology for a sparse network: The edge probability between two sensors is labeled with probability 0.95 if their distance is at most 1, and with probability 0.5 if their distance is larger than 1 and less than 2. Otherwise, there is no edge between the two sensors. In this topology, each sensor (i,j) that is not on or near the boundary of the grid generally has eight neighbors.
- A topology for a dense network: The edge probability between two sensors is labeled with probability 0.95 if their distance is at most 1.5, and with probability 0.5 if their distance is larger than 1.5 and less than 3. Otherwise, there is no edge between the two sensors. In this topology, each sensor (i,j) that is not on or near the boundary of the grid generally has 24 neighbors.

The performance of a flood sequencing protocol can be measured by the following two metrics:

1. *Reach:* The percentage of sensors that receive a message sent by sensor 0.
2. *Communication:* The total number of messages forwarded by all sensors in the network.

In our simulations, we do not consider other techniques that can improve the performance of a flood protocol based on extra information such as probability, location, and neighbor information.

TABLE 3
Performance of the Sequencing Free and Linear Sequencing Protocols

	sparse 10*10			sparse 20*20			dense 10*10			dense 20*20		
	hmax	Reach	Com.	hmax	Reach	Com.	hmax	Reach	Com.	hmax	Reach	Com.
free	13	99%	351.3	27	99.2%	2885.7	7	99.8%	200.5	13	99%	1262
lin	15	98.5%	97.8	28	98.5%	390.3	7	98.5%	87.5	14	98.8%	376.4

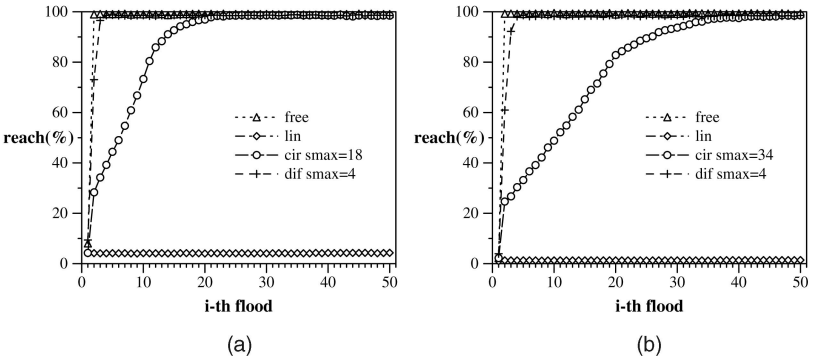

Fig. 4. Reach of the four flood sequencing protocols starting from an illegitimate state in sparse networks. (a) A 10^*10 network. (b) A 20^*20 network.

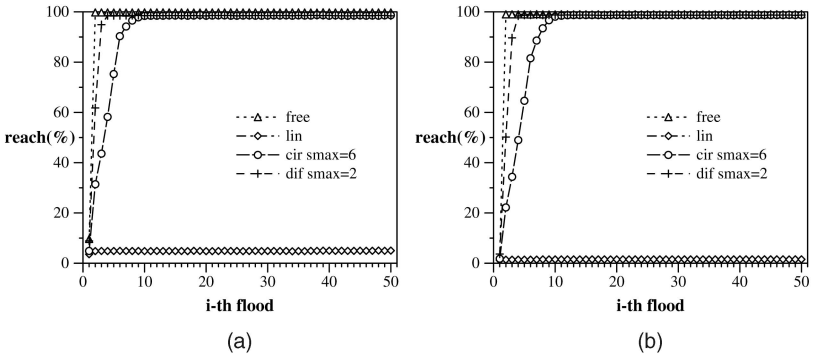

Fig. 5. Reach of the four flood sequencing protocols starting from an illegitimate state in dense networks. (a) A 10^*10 network. (b) A 20^*20 network.

9.2 Simulation Results

We first studied the performance of the sequencing free and linear sequencing protocols starting from a legitimate state. The result of each simulation in this study represents the average value over the simulations of 100,000 floods. Staring from a legitimate state, the linear sequencing protocol never discards fresh messages and never accepts redundant messages under any degree of message loss, and so we consider its performance as the ideal one for flood sequencing protocols that attach a sequence number to a flood message. When the value of $smax$ is reasonably large for a given network setting, the performance of the circular sequencing and differentiated sequencing protocols becomes the same as that of the linear sequencing protocol in terms of reach and communication.

Table 3 shows the reach and communication of the sequencing free and linear sequencing protocols in sparse and dense networks. Also the value of $hmax$ used in each network setting is specified in the table. Notice that the value of $hmax$ used in each dense network is around half of that used in its corresponding sparse network. In these simulations, $tmax = 6$ was used for a sparse network, and $tmax = 7$ was used for a dense network. From the above results, one can observe that the sequencing free protocol requires the sensors to send much more messages than those that the linear sequencing protocol does. Specially in the sparse $20 * 20$ network where a large value needs to be selected for $hmax$ $(= 27)$, the communication of the sequencing free protocol is around 7.39 times that of the linear sequencing protocol.

Next, we studied the stabilization properties of the four sequencing protocols, and their performance while stabilizing. We simulated the sequences of floods starting from 1,000 different illegitimate states, and computed the average reach for each ith flood. For the linear sequencing protocol that requires sensors to use unbounded sequence numbers, we simulated the protocol such that sensors use unsigned 16-bit integers, that have the range of 0 .. 65,535, for sequence numbers, and an initial value of $slast$ for each sensor is randomly selected from the range. Note that an initial value for $slast.0$ was selected such that $slast.0$ will not wrap around during the simulation of the first 100 floods. For the circular and differentiated sequencing protocols, we attempted to select an appropriate value of $smax$ for each network setting such that the assumption of bounded message loss becomes acceptable, while the convergence time of each protocol is minimized.

Figs. 4 and 5 show the reach of the four protocols starting from an illegitimate state in sparse networks and in dense

TABLE 4
Convergence Time in Analysis and in Simulation (Time Units)

	free		lin		cir		dif	
	Anal.	Sim.	Anal.	Sim.	Anal.	Sim.	Anal.	Sim.
sparse 10*10	$2 * f$	f	unbounded	unbounded	$20 * f$	$21 * f$	$4 * f$	$3 * f$
sparse 20*20	$2 * f$	f	unbounded	unbounded	$36 * f$	$40 * f$	$4 * f$	$3 * f$
dense 10*10	$2 * f$	f	unbounded	unbounded	$8 * f$	$9 * f$	$3 * f$	$3 * f$
dense 20*20	$2 * f$	f	unbounded	unbounded	$8 * f$	$9 * f$	$3 * f$	$3 * f$

networks, respectively. The behaviors of the protocols while stabilizing are discussed next.

In all the simulated settings, the reach of the first flood is very low. This is because forwarded messages initiated by sensor 0 and other sensors illegitimately collide with each other.

The sequencing free protocol converges to a legitimate state quickly. However, the communication of the protocol is much higher than that of the other protocols. The average communication of the protocol was 344.7 for the 10^*10 sparse network, 2,830.8 for the 20^*20 sparse network, 197.1 for the 10^*10 dense network, and 1,240.1 for the 20^*20 dense network.

The performance of the linear sequencing protocol starting from an illegitimate state is bad. The maximum difference $slast.u - slast.0$ for any sensor u in a network can be very large (in simulation, the maximum difference is 65,535), and the sensors that have $slast$ larger than $slast.0$ will not forward a received message, resulting in a low reach of the protocol.

The convergence of the circular sequencing protocol is affected by a topology type and network size. Moreover, these factors affect selecting a value for $smax$. The circular sequencing protocol converges faster to a legitimate state in a dense network than in a sparse network, since each sensor has a higher probability to receive a (fresh) flood message from one of its neighbors in a dense network than that in a sparse network. On the other hand, the convergence of the differentiated sequencing protocol is not affected by these factors, and a small value can be selected for $smax$, regardless of the factors.

In the circular and differentiated sequencing protocols, at the beginning, a flood message does not reach most of sensors in the network, since a sensor u, near sensor 0, that receives the message discards it, due to the wrong value of $slast.u$. As sensor 0 initiates more floods, a flood message can reach more sensors. The reach of the circular protocol increases slowly compared to that of the differentiated protocol. This is because each sensor has a higher probability to accept a received fresh message in the differentiated protocol (where it accepts a message if the message has a different sequence number than its last sequence number) than that in the circular protocol (where it accepts a message if the message has a logically larger sequence number than its last sequence number). In the circular protocol, after a flood message is propagated all over the network, some sensors still discard a received message (due to the wrong value of $slast$ stored in them), and then the reach increases more slowly until the convergence. Thus, in all the simulated network settings, the differentiated protocol reaches a legitimate state faster than the circular protocol does.

In Table 4, we compare their stabilization properties in simulation with those in analysis (as in Table 1). We measured the convergence time of each protocol such that the sequencing free protocol converges to a legitimate state if its reach becomes around the average reach of the protocol starting from a legitimate state, and the other protocols converge to a legitimate state if their reach becomes around the average reach of the linear sequencing protocol starting from a legitimate state. In the circular protocol, the convergence time measured in simulation becomes larger than that analyzed under the assumption of bounded message loss, while in the differentiated protocol, the convergence time measured in simulation is the same as or less than that analyzed under the same assumption. This is because it is harder to satisfy the assumption in the circular protocol, specially for a large sparse network. In some settings, the convergence time in simulation is smaller than that in analysis, since the convergence time in analysis is computed based on the worst case.

In summary, starting from any legitimate state, the performance of any flood sequencing protocol that attaches a sequence number to a flood message is better than that of the sequencing free protocol in terms of communication. Starting from an illegitimate state, the differentiated sequencing protocol converges to a legitimate state quickly in all the simulated network settings. Thus, we conclude that the differentiated sequencing protocol has better stabilization property and performance compared to those of the other three protocols.

10 Discussion

Frequent floods. The flood period, $hmax * tmax + 1$, used in previous sections is based on the lower bound on the flood period in Theorem 0. Recall that this bound is computed to guarantee that no two consecutive flood messages ever collide with each other. In a typical execution of the protocol, each sensor chooses its forwarding period at random in the range $1..tmax$, and so most sensors likely receive the flood messages within $(hmax - 1) * tmax/2$ time units, instead of $(hmax - 1) * tmax$ time units. Thus, in practical setting, sensor 0 may not need to wait a full period that guarantees no collision between two consecutive floods to initiate a next flood. When a shorter flood period is used, sensor 0 can flood a message frequently. However, in this case, multiple concurrent floods can exist in the network, which may affect the performance and stabilization property of a flood

TABLE 5
Performance of the Differentiated Sequencing Protocol with Different Flood Periods

Number of max concurrent floods	sparse 10*10			sparse 20*20			dense 10*10			dense 20*20		
	f	Reach	Com.	f	Reach	Com.	f	Reach	Com.	f	Reach	Com.
1	91	98.53%	97.85	169	98.46%	390.2	50	98.51%	87.44	99	98.73%	376.36
2	46	98.52%	97.84	85	98.45%	390.18	25	98.50%	87.44	50	98.73%	376.27
4	23	98.47%	97.87	43	98.45%	390.15	13	93.88%	81.96	25	98.70%	376.2
8	12	79.70%	85.18	22	94.95%	391.63	7	51.36%	42.64	13	71.19%	290.09

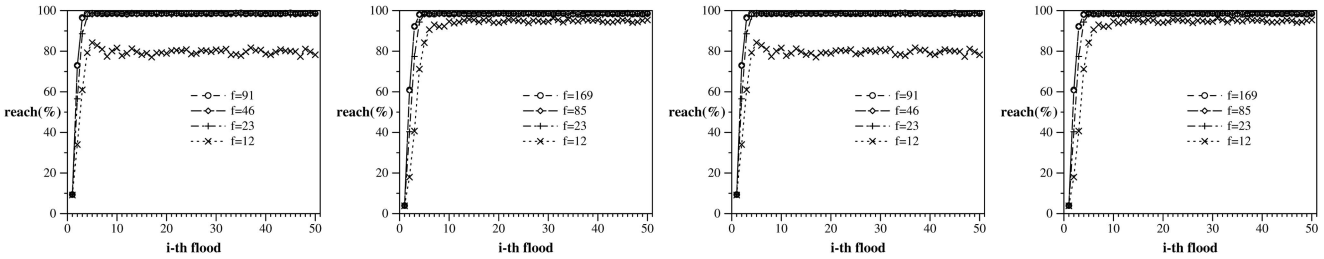

Fig. 6. Reach of the differentiated sequencing protocol, with different flood periods, starting from an illegitimate state. (a) A 10^*10 sparse network. (b) A 20^*20 sparse network. (c) A 10^*10 dense network. (d) A 20^*20 dense network.

sequencing protocol. We discuss the effect of frequent floods on the differentiated sequencing protocol.

We simulated the differentiated sequencing protocol such that sensor 0 initiates a flood frequently in the same network settings and with the same parameter values as in the previous section. Four different flood periods, $hmax * tmax + 1$, $(hmax * tmax + 1)/2$, $(hmax * tmax + 1)/4$, and $(hmax * tmax + 1)/8$ were used. In this study, we call the flood period of $hmax * tmax + 1$ a *full flood period*.

Table 5 shows the performance of the differentiated sequencing protocol, with different flood periods (f in time units), starting from a legitimate state. The result of each simulation in the table represents the average value over the simulations of 100,000 floods. In all the simulated settings, the performance of the protocol using half the full flood period remains the same as that using the full flood period. The performance using one fourth of the full flood period remains almost the same as that using the full flood period, except in the 10^*10 dense network where a relatively small flood period is used. The performance using one eighth of the full flood period is degraded. However, how much the performance is degraded is different depending on a network setting.

Fig. 6 shows the reach of the differentiated sequencing protocol, with different flood periods, starting from an illegitimate state in sparse and dense networks. We simulated the sequences of floods starting from 1,000 different illegitimate states, and computed the average reach for each ith flood. The stabilization property of the protocol remains (almost) the same in the 10^*10 networks. Using one fourth or one eighth of the full flood period, the stabilization property is degraded slightly in the 20^*20 networks.

In summary, in all the simulated settings, half the full flood period can be used without degrading the stabilization property and performance of the differentiated sequencing protocol. Even less than half the full flood period can be used, depending on a network topology type and size.

Multiple sources. In previous sections, only sensor 0 is a source that can initiate message floods. The flood sequencing protocols presented in this paper can be extended to support multiple sources. We discuss how the differentiated sequencing protocol can be extended such that each sensor can be a source.

If a large number of floods are forwarded concurrently, forwarded messages from these floods collide with one anther, causing many sensors not to receive any flood message. Thus, we assume that the number of floods that are initiated by different sensors and are forwarded concurrently is reasonably small. Each source needs to make sure that it waits f time units after initiating one flood and before initiating another flood, but after f time units, it can choose not to initiate a new flood.

Each flood message is augmented with a source ID as well as a sequence number. For a source w, each sensor maintains new, $slast$, and $hlast$ variables. When a sensor u receives a data(w, h, s) message, u accepts and forwards the message if s is different from $slast.w$. It is possible that u accepts another message from a different source w', before u times out and forwards the message from w. In this case, when u times out, u should send one composite message that consists of the two messages from w and w'. Also if u has a new message to flood, the new message is also combined into the composite message. Note that under the above assumption, a composite message will contain a small number of flood messages. When u receives a composite message, u processes each flood message in the composite message.

Starting from any state, the protocol converges to a legitimate state, provided that each source keeps initiating floods, and the assumption of bounded message loss in Section 7 is satisfied for each source.

11 Concluding Remarks

In this paper, we discussed a family of the four flood sequencing protocols that use sequence numbers to distinguish between fresh and redundant flood messages. The members of our family are the sequencing free protocol, the linear sequencing protocol, the circular sequencing protocol, and the differentiated sequencing protocol. We concluded that the differentiated sequencing protocol has better overall performance in terms of communication, and stabilization and stable properties, compared to those of the other three protocols. Note that our analysis is useful for sensor network designers or developers to select a proper flood sequencing protocol that satisfies the needs of a target sensor network.

A spanning tree can be used to distinguish between fresh and redundant flood messages [28], [29]. Flood protocols using a spanning tree require extra overheads to build and maintain the spanning tree. When sensors are mobile, the spanning tree needs to keep changed. Thus, these protocols may not be suitable for some sensor networks.

Appendix

The proof of Theorem 0. When sensor 0 broadcasts a data($hmax$, s) message at time t, an out-neighbor u of sensor 0 can receive the message at t and choose the maximum possible value $tmax$ for the forwarding period. At time $t + tmax$, u forwards the message as data($hmax - 1$, s). Similarly, an out-neighbor u' of sensor u can receive the message at $t + tmax$ and choose $tmax$ for the forwarding period. This forwarding process continues until this message makes $hmax$ hops. Therefore, some sensor u can receive the last data(1, s) message at time $t + (hmax - 1) * tmax$ in the worst case. Thus, the flood period needs to be at least $(hmax - 1) * tmax + 1$ time units to guarantee that no forwarded messages from two consecutive floods collide with one anther. □

The proof of Theorem 1A. The protocol can start from a state where some sensor u ($\neq 0$) has wrong initial values, true for new and $hmax$ for $hlast$. In this case, u initiates a flood illegitimately. This flood will be terminated within f time units, since u will time-out within $tmax$ time units, and the maximum lifetime of a flood message is $(hmax - 1) * tmax$ time units as shown in the proof of Theorem 0. After all illegitimately initiated floods are terminated, $new.u$ for every sensor u always becomes false when $\mathrm{timer.u} = 1$. Consider a case that the last illegitimately initiated flood message is sent and received at t, and sensor 0 broadcasts a new message at t. In this case, in $(t - 1,\ t)$, $new.u$ for some sensor u in the network is true. In time unit $(t + f - 1,\ t + f)$, timer.0 becomes 1 again, $new.u$ for every sensor u is false, and so predicate (P1) holds. Thus, the protocol reaches a legitimate state within $2 * f$ time units, and continuously stays in legitimate states. □

The proof of Theorem 1B. It is straightforward, since every sensor accepts every received message and so it discards no fresh message. □

The proof of Theorem 1C. A sensor u can receive at most one message at each time instant. In the worst case, u can accept a redundant message at each time instant during the convergence time, and so the maximum number of redundant messages accepted by u until convergence is $2 * f$. □

The proof of Theorem 2A. Let s be the value of $slast.0$ at an initial state. After f time units, any illegitimately initiated flood will be terminated, and $new.u$ for every sensor u always becomes false when $\mathrm{timer}.0 = 1$. Then, the value of timer.0 becomes 1 again, say in $(t - 1,\ t)$, with $2 * f$ time units (as shown in the proof of Theorem 1A). In $(t - 1,\ t)$, $slast.0$ is at least $s + 1$. The protocol has two cases to consider. Note that k is the maximum value between 1 and k', where k' is the maximum difference $slast.u - s$ for any u at the initial state. In the first case, at the initial state, s is larger than or equal to every $slast.u$ (i.e., $k' \leq 0$), or k' is one, and so $k = 1$. In this case, in $(t - 1,\ t)$, for every sensor u, $new.u$ is false, and $slast.u$ is at most $slast.0$ ($slast.u \leq slast.0$). Thus, predicate (P2) holds within $2 * f$ time units.

In the second case, at the initial state, the maximum difference $slast.u - s$ for any sensor u is larger than one (i.e., $k' > 1$), and so $k > 1$. In this case, in $(t - 1,\ t)$, $slast.0$ is less than some $slast.u$ in the network, and at t, sensor 0 broadcasts a message with sequence number s' where $s' \geq s + 2$. At $t + (k - 2) * f$, sensor 0 broadcasts a message with sequence number $s' + k - 2$ ($\geq s + k$) and this flood will be terminated by $t + (k - 1) * f - 1$. In time unit $(t + (k - 1) * f - 1,\ t + (k - 1) * f)$, for every sensor u, $new.u$ is false, and $slast.u$ is at most $slast.0$ ($slast.u \leq slast.0$). Thus, predicate (P2) holds within $(k + 1) * f$ time units. Thus, the protocol reaches a legitimate state within $(k + 1) * f$ time units, and continuously stays in legitimate states. □

The proof of Theorem 2B. A sensor u can receive at most one message at each time instant. In the worst case, sensor u can discard a fresh message at each time instant during the convergence time, and so the maximum number of fresh messages discarded by sensor u until convergence is $(k + 1) * f$. □

The proof of Theorem 2C. The protocol can start from a state where $new.u$ for every sensor u is true. In this case, every sensor u can initiate a flood of the previous accepted message. Thus, a sensor u can accept at most $n - 1$ redundant messages from every other sensor in the network. □

The proof of Theorem 3A. After f time units, any illegitimately initiated flood will be terminated, and $new.u$ for every sensor u always becomes false when $\mathrm{timer}.0 = 1$. The value of timer.0 becomes 1 again, say in $(t - 1,\ t)$, within $2 * f$ time units (similar to the proof of Theorem 1A), and then sensor 0 broadcasts a flood message at t. By the assumption of bounded message loss, every sensor u is guaranteed to receive at least one (fresh) flood message by $t + \frac{smax}{2} * f - 1$. When u receives a message, u computes whether the message is fresh or redundant based on the values of the received sequence number and $slast.u$. Because of message loss and/or

wrong initial value of $slast.u$, u may compute that the received message is redundant. Assume that in $(t + \frac{smax}{2} * f - 1,\ t + \frac{smax}{2} * f)$, the value of $slast.0$ is s and the value of $slast.u$ for some sensor u is equal to $(s + \frac{smax}{2}) \bmod (smax + 1)$. At $t + (smax - 1) * f$, sensor 0 broadcasts a message with sequence number $(s + \frac{smax}{2}) \bmod (smax + 1)$, and this flood message will be terminated by $t + smax * f - 1$. In $(t + smax * f - 1,\ t + smax * f)$, timer.0 becomes 1 again, and for every sensor u, $new.u$ is false, and $slast.0$ is logically larger than or equal to $slast.u$. Thus, predicate (P3) holds. Therefore, the protocol reaches a legitimate state within $(smax + 2) * f$ time units, and continuously stays in legitimate states. □

The proof of Theorem 3B. This proof is similar to that of Theorem 2B.

The proof of Theorem 3C. During the first f time units, some illegitimately initiated floods can exist in the network. During this period, a sensor u can accept at most f redundant messages, since u can receive and accept a redundant message at each time unit in the worst case. After all illegitimately initiated floods are terminated, any forwarded message is initiated by sensor 0. Assume that sensor 0 initiates a flood with s at $t - 2$, and all illegitimately initiated floods are terminated at t. Also assume that a sensor u accepts the message with s at $t - 2$, an illegitimately initiated message with $s' = (s + \frac{smax}{2}) \bmod (smax + 1)$ (which is logically larger than s) at $t - 1$, and another illegitimately initiated message with $s'' = (s - 1) \bmod (smax + 1)$ (which is logically larger than s') at t. At $t + 1$, if u receives the same message with s, u accepts it again, since s is logically larger than s''. Then, u will not accept any redundant message any more. Thus, the maximum number of redundant messages accepted by sensor u is $f + 1$. □

The proof of Theorem 4A. Similar to the proof of Theorem 3A, after all illegitimately initiated floods are terminated (within f time units), the value of timer.0 becomes 1 again, say in $(t - 1,\ t)$, within $2 * f$ time units. Assume that sensor 0 broadcasts a new message with sequence number s at t. Then, sensor 0 broadcasts a new message with sequence number $(s + \frac{smax}{2} - 1) \bmod (smax + 1)$ at $t + (\frac{smax}{2} - 1) * f$. In time unit $(t + \frac{smax}{2} * f - 1,\ t + \frac{smax}{2} * f)$, timer.0 becomes 1 again, and for every sensor u, $new.u$ is false, and $slast.u$ has one of the values in $s\ ..\ (s + \frac{smax}{2} - 1) \bmod (smax + 1)$, since u receives at least one of those sequence numbers by the assumption of bounded message loss. Thus, predicate (P4) holds. Therefore, the protocol reaches a legitimate state within $(\frac{smax}{2} + 2) * f$ time units, and continuously stays in legitimate states. □

The proof of Theorem 4B. This proof is similar to that of Theorem 2B.

The proof of Theorem 4C. Similar to the proof of Theorem 3C, during the first f time units, a sensor u can accept at most f redundant messages. Assume that sensor 0 initiates a flood with s at $t - 1$, and all illegitimately initiated floods are terminated at t. Also assume that a sensor u accepts the message with s at $t - 1$, and an illegitimately initiated message with s' ($\neq s$) at t. At $t + 1$, if u receives the same message with s, u accepts it again, since $s' \neq s$. Then, u will not accept any redundant message any more. Thus, the maximum number of redundant messages accepted by u is $f + 1$. □

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation (NSF) under Grant No. 0520250. The authors are grateful to anonymous referees for their helpful comments. A preliminary version of this paper appeared at the Ninth International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2007) [1]. A part of this work was done while Choi was with the Department of Computer Sciences, University of Texas at Austin, Austin, Texas.

REFERENCES

[1] Y. Choi and M. Gouda, "Stabilization of Flood Sequencing Protocols in Sensor Networks," *Proc. Ninth Int'l Symp. Stabilization, Safety, and Security of Distributed Systems (SSS '07)*, Nov. 2007.

[2] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *Proc. ACM MobiCom*, pp. 151-162, 1999.

[3] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic Broadcast for Flooding in Wireless Mobile Ad Hoc Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC)*, 2003.

[4] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14-25, Jan. 2002.

[5] W. Lou and J. Wu, "On Reducing Broadcast Redundancy in Ad Hoc Wireless Networks," *IEEE Trans. Mobile Computing*, vol. 1, no. 2, pp. 111-122, Apr.-June 2002.

[6] J. Li and P. Mohapatra, "A Novel Mechanism for Flooding Based Route Discovery in Ad Hoc Networks," *Proc. IEEE Global Telecomm. Conf.*, 2003.

[7] B. Williams and T. Camp, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, 2002.

[8] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, Chapter 5, vol. 353, pp. 153-181, Kluwer Academic Publishers, 1996.

[9] M. Sun, W. Feng, and T. Lai, "Location Aided Broadcast in Wireless Ad Hoc Networks," *Proc. IEEE Global Telecomm. Conf.*, pp. 2842-2846, Nov. 2001.

[10] H. Lim and C. Kim, "Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks," *Proc. ACM Int'l Workshop Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, 2000.

[11] W. Peng and X. Lu, "AHBP: An Efficient Broadcast Protocol for Mobile Ad Hoc Network," *J. Science and Technology*, 2001.

[12] J. Wu and F. Dai, "Broadcasting in Ad Hoc Networks Based on Self-Pruning," *Proc. IEEE INFOCOM*, 2003.

[13] A. Durresi, V. Paruchuri, S.S. lyengar, and R. Kannan, "Optimized Broadcast Protocol for Sensor Networks," *IEEE Trans. Computer*, vol. 54, no. 8, pp. 1013-1024, Aug. 2005.

[14] H. Sabbineni and K. Chakrabarty, "Location-Aided Flooding: An Energy-Efficient Data Dissemination Protocol for Wireless-Sensor Networks," *IEEE Trans. Computers*, vol. 54, no. 1, pp. 36-46, Jan. 2005.

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

[16] M. Heissenbüttel, T. Braun, M. Waelchli, and T. Bernoulli, "Optimized Stateless Broadcasting in Wireless Multi-Hop Networks," *Proc. IEEE INFOCOM*, 2006.

[17] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 784-803, Dec. 1997.

[18] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," *Wireless Networks,* vol. 8, nos. 2/3, pp. 169-185, 2002.
[19] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom,* 2000.
[20] M. Gouda and Y. Choi, "A State-Based Model of Sensor Protocols," *Proc. Ninth Int'l Conf. Principles of Distributed Systems (OPODIS '05),* Dec. 2005.
[21] "Tinyos," http://www.tinyos.net, 2009.
[22] Y. Choi, M.G. Gouda, M.C. Kim, and A. Arora, "The Mote Connectivity Protocol," *Proc. 12th Int'l Conf. Computer Comm. and Networks (ICCCN '03),* pp. 533-538, Oct. 2003.
[23] A. Woo, T. Tony, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," *Proc. ACM SenSys,* 2003.
[24] A. Cerpa, N. Busek, and D. Estrin, "SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments," CENS Technical Report 21, Sept. 2003.
[25] F. Kuhn and A. Zollinger, "Ad-Hoc Networks beyond Unit Disk Graphs," *Proc. 2003 Joint Workshop Foundations of Mobile Computing: Discrete Algorithms and Methods for Mobile Computing and Comm.-Principles of Mobile Computing (DIALM-POMC '03),* pp. 69-78, 2003.
[26] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing Newly Deployed Ad Hoc and Sensor Networks," *Proc. ACM MOBICOM '04,* pp. 260-274, 2004.
[27] L. Barriére, P. Fraigniaud, and L. Narayanan, "Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges," *Proc. Fifth Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm. (DIALM '01),* pp. 19-27, 2001.
[28] B. Bellur and R.G. Ogier, "A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks," *Proc. IEEE INFOCOM,* pp. 178-186, 1999.
[29] T. Korkmaz and M. Krunz, "Hybrid Flooding and Tree-Based Broadcasting for Reliable and Efficient Link-State Dissemination," *Proc. IEEE GLOBECOM '02 Conf.—High-Speed Networks Symp.,* Nov. 2002.

Young-ri Choi received the BS degree in computer science from Yonsei University, Seoul, Korea, in 1998. She received the MS and PhD degrees in computer sciences from the University of Texas at Austin in 2002 and 2007, respectively. Her research interests include network protocols, sensor networks, secure computing, fault tolerance, and distributed computing. She is currently with Korea Institute of Science and Technology Information, Daejeon, Republic of Korea.

Chin-Tser Huang received the BS degree in computer science and information engineering from the National Taiwan University, Taipei, in 1993, and the MS and PhD degrees in computer sciences from the University of Texas at Austin in 1998 and 2003, respectively. He joined the faculty at the University of South Carolina in Columbia in 2003 and is now an assistant professor in the Department of Computer Science and Engineering. His research interests include network security, network protocol design and verification, and distributed systems. He is the director of the Secure Protocol Implementation and Development (SPID) Laboratory at the University of South Carolina. He is the author (along with Mohamed Gouda) of the book *Hop Integrity in the Internet*, published by Springer in 2005. He is a member of the Sigma Xi, the Upsilon Pi Epsilon, the IEEE, the IEEE Computer Society, and the ACM.

Mohamed G. Gouda received the BS degrees in engineering and mathematics from Cairo University in 1968 and 1971, respectively, the MA degree in mathematics from York University, Ontario, Canada, in 1972, and the master's and PhD degrees in computer science from the University of Waterloo, Ontario, Canada, in 1973 and 1977, respectively. He currently holds the Mike A. Myers centennial professorship in computer sciences at the University of Texas at Austin. His research areas are distributed and concurrent computing and network protocols. In these areas, he has been working on abstraction, formality, correctness, nondeterminism, atomicity, reliability, security, convergence, and stabilization. He has published more than 15 book chapters, more than 60 journal papers, and more than 90 conference and workshop papers. He is the author of the textbook *Elements of Network Protocol Design* (John Wiley and Sons, 1998). This is the first ever textbook where network protocols are presented in an abstract and formal setting. He coauthored, with Tommy M. McGuire, the monograph *The Austin Protocol Compiler* (Springer, 2005). He also coauthored, with Chin-Tser Huang, the monograph *Hop Integrity in the Internet* (Springer, 2006). He is the 1993 winner of the Kuwait Award in Basic Sciences. He is also the recipient of the IBM Faculty Partnership Award for the academic year 2000-2001 and again for the academic year 2001-2002 and he became a fellow of the IBM Center for Advanced Studies in Austin in 2002. He won the 2001 IEEE Communication Society William R. Bennet Best Paper Award. In 2004, his paper "Diverse Firewall Design," coauthored with Alex X. Liu, won the William C. Carter Award. He is a member of the IEEE and the IEEE Computer Society. More details about his research and background can be found at http://www.cs.utexas.edu/users/gouda.