Disciplined Flood Protocols in Sensor Networks

Young-ri Choi
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-0233, U.S.A.
yrchoi@cs.utexas.edu

Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-0233, U.S.A.
gouda@cs.utexas.edu

UTCS Technical Report TR-05-03

Abstract

Flood is a communication primitive that can be initiated by the base station of a sensor network to send a copy of some message to every sensor in the network. When a flood of some message is initiated, the message is forwarded by every sensor that receives the message until the sensors decide not to forward the message any more. This uncontroled flood can cause the forwarded messages to collide with one another, with the result that many sensors in the network do not receive any copy of the flooded message. In this paper, we present a family of flood protocols, called the disciplined flood protocols, that aim to reduce or prevent most message collisions that occur in a regular flood protocol. We show by simulation that whereas a regular flood protocol can cause a flooded message to reach between 60% and 80% of all sensors in the network, a disciplined flood protocol can cause a flooded message to reach between 88% and 99% of all sensors in the network.

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 data message to every sensor in the network. The execution of a flood starts by the base station sending a copy of the data message to everyone of its neighboring sensors. Whenever a sensor receives a data message, it keeps a copy of the message and forwards the message to everyone of its neighboring sensors and the cycle repeats.

To limit the (potentially indefinite) forwarding of a flooded data message within a sensor network, the message is augmented with a field h whose value is in a range $0..hmax$. When the base station sends the data message for the first time, field h in the message has the value $hmax$. Whenever a sensor receives a data(h) message, where $h \geq 1$, then the sensor forwards the message as a data($h-1$) message. Whenever a sensor receives a data(0) message, then the sensor does not forward the message any further.

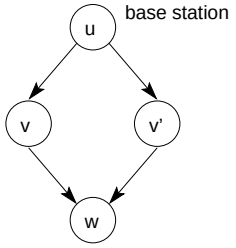


Figure 1. A sensor network

Flood has several significant uses in sensor networks. In one use, the base station of a sensor network needs to reset the network, and it uses flood to send a reset message to every sensor in the network requesting that each sensor resets itself upon receiving the message. In a second use, the base station needs to pass some data message to some (not necessarily all) sensors in the network. In this case, the base station uses flood to send the data message to all the sensors in the network, but name in the message those sensors that should find the message relevant.

Unfortunately, flood can cause severe problems in sensor networks.

i. *Collisions within a Single Flood:* Consider the sensor network in Figure 1. If the base station in this network initiates a flood by sending a data(1) message, then each of the two neighboring sensors, v and v', of the base station receives the message and forwards it as a data(0) message. Because the two data(0) messages are forwarded (by v and v') at the same time, they collide and sensor w never gets a copy of the flooded data

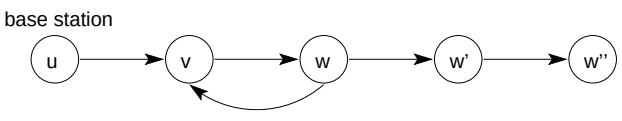


Figure 2. A sensor network

message.

ii. *Collisions of Consecutive Floods:* If the base station of a sensor network 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.

iii. *Redundant Forwarding:* Consider the sensor network in Figure 2. If the base station in this network initiates a flood by sending a data(3) message, then
sensor v forwards a data(2) message,
sensor w forwards a data(1) message,
sensor w' forwards a data(0) message, and
sensor w'' forwards no message.
Unfortunately sensor v also receives the data(1) message that is forwarded by sensor w and redundantly forwards it as a data(0) message. Thus, both sensors v and w redundantly receive the message one time each, and sensor v redundantly forwards the message one more time.

A common method to recognize redundantly forwarded messages is to attach a sequence number to each flood message as in [4], [3]. In a sensor network, these sequence numbers should cover a small range, since sensors have limited memory and bandwidth. In this case, a sensor may fail to distinguish new messages from redundantly forwarded messages, if some flood messages are lost.

In this paper, we present a family of flood protocols where the above three problems do not occur. We refer to the protocols in this family as disciplined flood protocols. A disciplined flood protocol has the following important features or properties:

i. *Few Collisions within a Single Flood:* When a sensor u receives a data(h) message and checks that $h \geq 1$ (which means that u needs to forward the message), then u selects a random time period, called the forwarding period, and forwards the message only at the end of that period. (Recall that u forwards the message as a data($h-1$) message.)

ii. *No Collisions of Consecutive Floods:* After the base station initiates a flood by sending a data($hmax$) message, it abstains from initiating a second flood for a long enough time period until it is certain that the sensors in the network are no longer forwarding data messages that belong to the first flood. In this paper, we refer to the time period between two consecutive floods as the flood period, and compute a lower bound on the flood period that can be used in our disciplined flood protocols.

iii. *No Redundant Forwarding:* When a sensor u receives a data(h) message and decides that it needs to forward the message as a data($h-1$) message after a random forwarding period, u computes a time period called the deafness period. If u receives any data(h') message during the computed deafness period, u concludes that the data(h') message belongs to the same flood as that of the earlier data(h) message, and discards the data(h') message without keeping a copy of it and without forwarding it.

Several flood protocols have been proposed to reduce the redundantly forwarded messages in a flood based on probability, location, or neighbor information [4], [5], [6], [7]. Unlike these protocols, our disciplined flood protocols control the activities within a single flood or across consecutive floods in order to satisfy the above three properties.

2. A Model of Sensor Networks

In this section, we present a formal model of the execution of a sensor network. We use this model to specify the disciplined flood protocol in the next section. We also use this model to verify this protocol in Section 4, and to develop our simulation in Section 5.

The *topology* of a sensor network is a directed graph that satisfies the following two conditions. First, each node in the topology represents a distinct sensor in the sensor network. Second, each directed edge (u, v) from node u to node v in the topology indicates that every message that is sent by sensor u can be received by sensor v (provided that neither sensor v nor any "neighboring sensor" of v sends a message at the same time when sensor u sends its message).

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. (Note that a sensor can be both an in-neighbor and an out-neighbor of another sensor in the sensor network.)

As an example, Figure 3 shows the topology of a sensor network. This network has six sensors, and sensor u in this network has three out-neighbors, namely sensors v, v', and v''. Thus, if sensor u sends a message, then this message can be received simultaneously by the three sensors v, and v', and v''. Note that sensor u is both an in-neighbor and out-neighbor of sensor v' in this network.

We assume that during the execution of a sensor network, the real-time passes through discrete instants: instant 1, in-

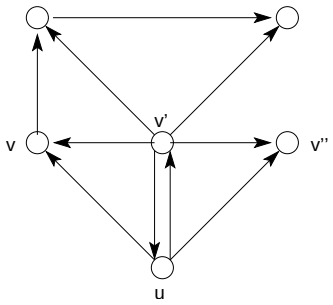


Figure 3. Topology of a sensor network

stant 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 as a *time unit*. (The value of a time unit is not critical to our current presentation of a sensor network model, but we estimate that the value of the time unit is around 100 milliseconds.)

At a time instant t, the time-out of a sensor u may expire causing u to execute its timeout action. Executing the timeout action of sensor u causes u to update its own local variables and to send at most one message. It may also cause 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 instant t. The timeout action of sensor u is of the following form:

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

To keep track of its time-out, each sensor u has an implicit variable named "timer.u". At each time instant, variable timer.u is either "present" or "not-present". Moreover, if variable timer.u is present at an instant t, then it has a positive integer value at t. Otherwise, it is not-present and has no value at t.

If sensor u executes a statement "timeout-after <expression>" at instant t, then timer.u is present at $t+1$ and its value at $t+1$ is the value of <expression> at instant t.

If timer.u is present and its value is k, where $k > 1$, at instant t, then timer.u is present and its value is $k-1$ at instant $t+1$.

If timer.u is present and its value is 1 at t, then sensor u executes its timeout action at t and timer.u is not-present at $t+1$ unless u executes "timeout-after <expression> as part of its timeout action.

If a sensor u executes its timeout action and sends a message at instant t, then any sensor v, that is an out-neighbor of u, receives a copy of the message at instant t, provided that the following two conditions hold.

i. Sensor v does not send any message at instant t. (This condition indicates that either sensor v does not execute its timeout action at t, or it executes its timeout action at t but this execution of its timeout action does not include sending a message.)

ii. Sensor v has no in-neighbor, other than sensor u, that sends a message at instant t. (If v sends a message at t or if an in-neighbor of v, other than u, sends a message at t, then this message is said to *collide* with the message sent by u at t with the net result that v receives no message at t.)

If a sensor u receives a message at time 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 and it may cause u to execute the statement "timeout-after <expression>" which causes the time-out of u to expire after k time units, where k is the value of <expression> at instant t. The receiving action of sensor u is of the following form:

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

It follows from the above discussion that at a time instant, a sensor u executes exactly one of the following:

i. u sends one message, but receives no message.

ii. u receives one message, but sends no message.

iii. u sends no message and receives no message.

In the next section, we specify the discipline flood protocol using the formal model of sensor protocols in this section.

3. Disciplined Flood 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 sends a message of the form data($hmax$), where $hmax$ is the number of hops to be made by this data message in the network.

Once sensor 0 broadcasts a message, it needs to wait enough time until this message is no longer forwarded in the network, before broadcasting a next message. The time period that sensor 0 needs to wait after broadcasting a message and before broadcasting a next message is called the

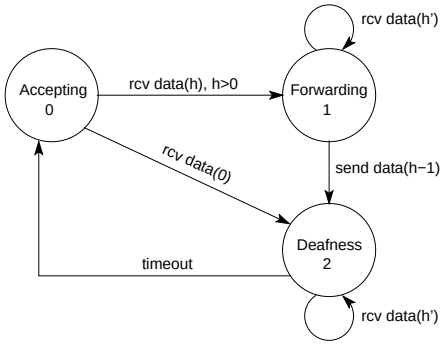


Figure 4. Three states of a sensor

flood period. The flood period consists of f time units. (A lower bound on the value of f is computed in the next section.) Thus, after sensor 0 broadcasts a message, it sets its timeout to expire after f time units in order to broadcast a next message.

A formal specification for sensor 0 is as follows.

```
sensor 0                  {base station}

const hmax    : integer, {max hop count}
      f       : integer  {flood period}
begin
   timeout-expires -> {generate new msg}
                      send data(hmax);
                      timeout-after f
end
```

Note that sensor 0 does not receive any messages.

When a sensor receives a data(h) message, the sensor accepts the message and forwards it as a data($h-1$) message, provided $h > 0$. To reduce the probability of message collision, any sensor u, that receives a message, chooses a random period whose length is chosen uniformly from the domain $1..tmax$, and sets its timeout to expire after the 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*.

Once sensor u accepts a received message and decides that it needs to forward the message after a random forwarding period, sensor u discards all subsequently received messages during a time period, called the *deafness period*, until the same message is no longer forwarded in the network. This way, sensor u is guaranteed not to accept or forward the same message multiple times. The deafness period consists of d time units. (A lower bound on the value of d is computed in the next section.) At the end of the deafness period, sensor u times-out and becomes ready to accept and forward the next received message.

At each instant, a sensor u is in any of three states: an accepting state, a forwarding state and a deafness state. In the accepting state, u is ready to accept and forward any data(h) message it receives. In the forwarding state, u is waiting for its forwarding period to finish, so that it can forward the last data(h) message it has accepted. In the deafness state, u discards any data(h) message it receives. It stays in this state for the duration of the deafness period (d time units). Figure 4 shows the three states of sensor u and the different transitions between them. The cycle of an accepting state followed by a forwarding state and then a deafness state (or an accepting state followed by a deafness state) is repeated over and over. Sensor u maintains a state variable st that has three possible values 0, 1, and 2.

$st = 0$ if u is in accepting state
1 if u is in forwarding state
2 if u is in deafness state

A formal specification for sensors $1..n-1$ is as follows.

```
sensor 1..n-1

const hmax    : integer, {max hop count}
      tmax    : integer, {max frwrding period}
      d       : integer  {deafness period}
var   st      : 0..2,    {state, init. 0}
      h,hlast : 0..hmax, {rcvd,last hop cnt}
      t       : 1..tmax  {forwarding period}
begin
   timeout-expires ->
           if st!=1 -> st := 0
           [] st=1  -> send data(hlast);
                       st := 2;
                       timeout-after d+1
           fi

[] rcv data(h) ->
   if st=0 -> {accept msg}
         if h>0 -> st := 1;
                   hlast := h-1;
                   t := random;
                   timeout-after t
         [] h=0 -> st := 2;
                   timeout-after d+1
         fi
   [] st>0 -> skip
   fi
end
```

4. Protocol Analysis

In this section, we estimate the deafness period and the flood period of the above disciplined protocol, and analyze the behavior of this protocol.

Theorem 1 : *(The Deafness Period Theorem)*

$$d \geq hmax * tmax$$

Proof: When sensor 0 broadcasts a data(hmax) message at time s, an out-neighbor u of sensor 0 receives it at s and can

choose the maximum possible value $tmax$ for the forwarding period. At time $s + tmax$, u sends it as a data(hmax-1) message. Similarly, an out-neighbor u' of sensor u receives it at $s + tmax$ and can choose $tmax$ for the forwarding period. At time $s + 2 * tmax$, u' sends it as a data(hmax-2) message. This forwarding process continues until this message makes $hmax + 1$ hops (i.e. $h = 0$). Therefore, some sensor u can receive the last data(0) message at time $s + hmax * tmax$ in the worst case. Based on this observation, the maximum time period a sensor u can receive the same message again, after u receives the message for the first time, is less than or equal to $hmax * tmax$. The deafness period needs be at least $hmax * tmax$ to guarantee that sensor u does not accept and forward the same message again. □

Theorem 2 : *(The Flood Period Theorem)*

$$f \geq hmax * tmax + d + 1$$

Proof When sensor 0 broadcasts a message at time s, some sensor u can receive the last data(0) message at time $s+hmax*tmax$ in the worst case. Then u stays in the deafness state for d time units, and finally u becomes ready to accept a next message at $s + hmax * tmax + d + 1$. Thus, the maximum time period from the time sensor 0 sends a message and to the time sensor u becomes ready to accept the next message is $hmax * tmax + d + 1$. The flood period needs to be at least $hmax * tmax + d + 1$ to guarantee that no forwarded messages from two floods collide with one anther, and every sensor is ready to accept a message when sensor 0 broadcasts a new message. □

To analyze the protocol, we use the minimum possible values for the flood period f and the deafness period d, from Theorem 1 and 2, as follows:

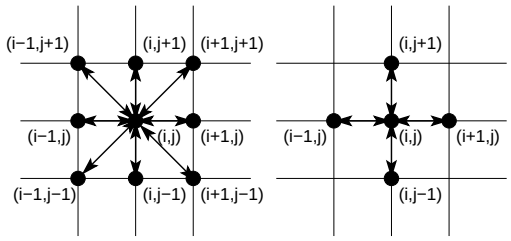
$$d = hmax * tmax$$
$$f = 2 * hmax * tmax + 1$$

Adopting these values of d and f, it is straightforward to show that the disciplined flood protocol satisfies the following three properties discussed in Section 1: (i) Few collisions within a single flood. (ii) No collisions of consecutive floods. (iii) No redundant forwarding.

Note that in this protocol, if a sensor receives a message in an accepting state, then the message is new and the sensor accepts it. Moreover, sensor 0 broadcasts a new message only when every sensor in the network is in an accepting state. Therefore, the sensor does not discard any new message, if the sensor receives it.

5. Protocol Simulation

We have developed a simulator that can simulate the execution of a regular flood protocol and the execution of our disciplined flood protocol. In this simulator, a network is an $N * N$ grid where N is the number of sensors in each side of the grid. This simulator allows us to configure the parameters of a protocol such as $tmax$ and $hmax$.



(a) A dense network (b) A scarce network

Figure 5. Topology

For the purpose of simulation, sensor 0 is (0,0) which is located at the left-bottom conner in a grid, and the following two types of a topology were used.

- A topology for a dense network: Each sensor (i,j) in a grid generally has eight (in- and out-) neighbors (i+1,j), (i+1,j+1), (i,j+1), (i-1,j+1), (i-1,j), (i-1,j-1), (i,j-1), and (i+1,j-1) as Figure 5(a).
- A topology for a scarce network: Each sensor (i,j) in a grid generally four (in- and out-) neighbors (i+1,j), (i,j+1), (i-1,j), and (i,j-1) as Figure 5(b).

Note that in most sensor networks, sensors are densely deployed. So we used the topology for a dense network in most simulations.

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

i. Reach: The percentage of sensors that receive a message sent by sensor 0.

ii. Frequency: The inverse of the length of the flood period which indicates how often sensor 0 can initiate the flooding of a new message.

iii. Latency: The average time it takes for a sensor to receive a message, after sensor 0 sends the message.

We ran simulations of a regular flood protocol and the disciplined flood protocol described in Section 3 in a 10*10 grid. Each simulation result in figures and tables represents the average value over 100 simulations. Note that in the

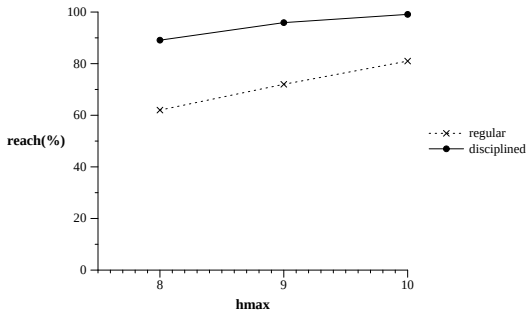


Figure 6. Reach of Regular Flood and of Disciplined Flood

dense network, the value of $hmax$ needs to be at least 8, so that a message broadcasted by sensor 0 can reach all sensors in the grid, while in the scarce network, the value of $hmax$ needs to be at least 17.

We studied the following scenarios for a regular flood, where a sensor forwards a message whenever it receives the message, until the message makes its maximum hops, in the dense network.

- When each sensor forwards a message at the next time unit from the instant that the sensor receives the message, the simulation result shows that only around 60% sensors in the network receive the flood message.
- When each sensor chooses a random forwarding period from the domain $1..tmax$, where $tmax = 10$, and waits for the forwarding period before sending a received message, the reach of the protocol can be increased up to 80% when $hmax = 10$. Note that the total number of messages sent by sensors in the network is increased as $hmax$ is increased.
- When sensor 0 broadcasts a second message (at time 25) shortly after it broadcasts the first message (at time 0), the reach of the second flood is very lower (31-38%) than that of the first flood (58-78%). The reason is because the forwarded messages for the second flood are collided with the forwarded messages for the first flood. Specially, if all forwarded messages for the second flood are collided in the first few hops, the reach of the second flood becomes very low.

Figure 6 shows the reach of the regular flood and of the disciplined flood when $tmax = 10$ in the dense network. In the regular flood, there is no detection of redundant forwarding, and so a message sent by a sensor for the first time can be collided with the redundantly forwarded messages. On the other hand, the disciplined flood has no redundant forwarding. Thus, the reach of the disciplined flood is higher than that of the regular flood.

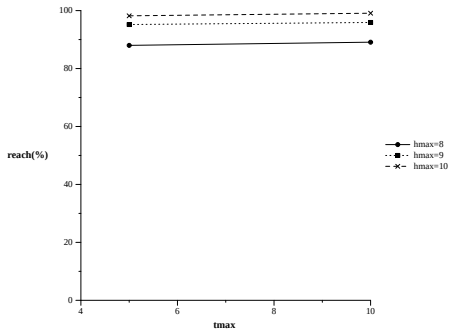


Figure 7. Reach of Disciplined Flood in a dense network vs. $tmax$

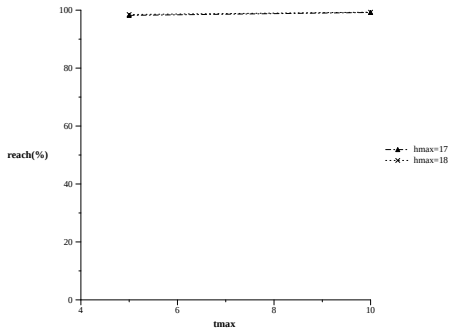


Figure 8. Reach of Disciplined Flood in a scarce network vs. $tmax$

Figure 7 shows the relationship between the value of $tmax$ and the reach of the disciplined flood protocol in the dense network, when $tmax$ =5 and 10. Figure 8 shows the relationship between the value of $tmax$ and the reach of the disciplined flood protocol in the scarce network, when $tmax$ =5 and 10. In both networks, if $tmax$ is 10 time units or more, then $tmax$ no longer affects the reach of the protocol any more in both networks.

In addition, the relationship between the value of $hmax$ and the reach of the protocol can be observed from Figures 7 and 8 as follows: In the dense network, as $hmax$ is increased, the reach of the protocol is increased, since the probability of message collision is high in the dense network, so additional few hops the message makes can increase the reach. On the other hand, in the scarce network, $hmax$ does not affect the reach of the protocol, since the probability of message collision is low.

Next we discuss the effect of $tmax$ and $hmax$ on the frequency and latency of the disciplined flood protocol in the dense network. Table 1 shows the flood period of the protocol over various $tmax$ and $hmax$ values. As $tmax$ and $hmax$ are increased, the flood period is increased, i.e.

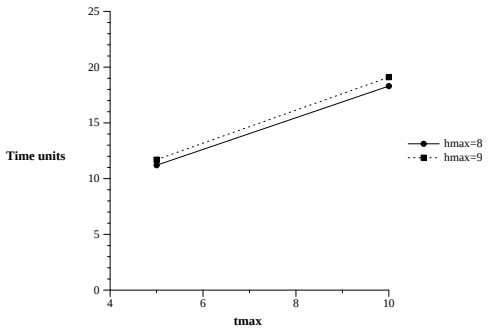


Figure 9. Latency of Disciplined Flood in a dense network

the frequency of the protocol is decreased. Thus, to have a high frequency, the protocol needs to have small $tmax$ and $hmax$ values.

tmax	hmax=8	hmax=9	hmax=10
5	81	91	101
10	161	181	201

Table 1. The flood period (in time units)

Figure 9 shows the effect of $tmax$ and $hmax$ on the latency of the disciplined flood protocol when $tmax$ =5 and 10, and $hmax$ =8 and 9 in the dense network. The latency of the protocol is increased linearly as $tmax$ is increased, but the latency does not depend on $hmax$. Thus, to have a low latency, the protocol needs to have a small $tmax$ value, regardless of the value of $hmax$.

When $tmax$ and $hmax$ have the large values, the flood period of the disciplined flood protocol becomes very long. (If $tmax = 10$ and $hmax = 10$, sensor 0 needs to wait for 20 seconds to broadcast a new message, assuming that a time unit is 100 milliseconds.) The flood period is computed to guarantee that no collisions of two consecutive floods happen, and every sensor is ready to accept a message when sensor 0 broadcasts a new message. In practical setting, a sensor chooses its forwarding period at random from the domain $1..tmax$. Therefore, most sensors in the network likely receive the flooded messages within $hmax * tmax/2$ time units, instead of $hmax * tmax$ computed in Theorem 2. Moreover, the flood of a message is affected by the topology of a network, and a flood message can be forwarded faster in some types of network topologies. Thus, one may use the half of the flood period, without significant performance degrading of the protocol. However, in this case, the protocol cannot guarantee that no collisions of two consecutive floods happen, and a sensor may fail to distinguish new messages from redundantly forwarded messages.

In the next section, we develop another version of the disciplined flood protocol in which one bit sequence number is added to each flood message. This allows us to double the frequency of the protocol, i.e. to reduce the flood period by a factor of two.

6. Multi-Flood Protocol

In this section, we discuss a second flood protocol where one bit sequence number of 0 and 1 is attached to each flood message and used to distinguish new messages from redundantly forwarded messages, unlike the disciplined flood protocol in Section 3. This new protocol is called the multi-flood protocol.

In the disciplined flood protocol, after a message broadcasted by sensor 0 makes the last hop, no message is forwarded in the network, and every sensor stays either in a deafness state or in an accepting state until the end of the flood period. This observation suggests one way to reduce the flood period as follows: Sensor 0 initiates the flooding of a new message if the previous message is no longer forwarded in the network, without waiting that all sensors finish their deafness periods (for the previous message). However, when sensor 0 broadcasts the new message, many sensors might be still in the deafness period and discard the received message that is new. To recognize the new message, each flood message is of the form:

`data(h,s)`

where field s is the sequence number of this message. The sequence number of any message only has two possible values 0 and 1.

When sensor 0 broadcasts a message, the message field s is assigned the toggled sequence number of the last message. (That is, if the last message has a sequence number 0, then this message has a sequence number 1, and vice versa.) If a sensor is in a deafness period for the message whose sequence number is 0, and receives a data($h, 1$) message, then it concludes that the flooding of the previous message with sequence number 0 is done, and this data($h, 1$) message is a new message. Similarly, if the sensor is in a deafness period for the message whose sequence number is 1, and receives a data($h, 0$) message, then it concludes that the flooding of the previous message with sequence number 1 is done, and this data($h, 0$) message is a new message.

Once sensor 0 broadcasts a message, sensor 0 needs to wait for the multi-flood period which is the flood period of this protocol. The multi-flood period consists of m time units. (The value of m can be the half of f time units in Section 3. A lower bound on the value of m is computed later in this section.) Thus, after sensor 0 broadcasts a message, it sets its timeout to expire after m time units to broadcast a next message.

A formal specification for sensor 0 is as follows.

```
sensor 0                  {base station}

const hmax    : integer, {max hop count}
      m       : integer  {multi-flood period}
var   slast   : 0..1     {last seq num}
begin
   timeout-expires -> {generate new msg}
              slast := (slast+1) mod 2;
              send data(hmax,slast);
              timeout-after m
end
```

When a sensor u receives a data(h, s) message, if sensor u is in an accepting state, it accepts the message, regardless of the sequence number of the message, and forwards it as a data($h-1$,s) message provided $h > 0$. If sensor u is in a deafness state, there are two cases to consider.

i. The sequence number of the last accepted message is different from s: In this case, u concludes that this message is a new message. Thus, u accepts the message, and forwards it provided $h > 0$.

ii. The sequence number of the last accepted message is the same as s: In this case, u concludes that this message belongs to the same flood as that of the last accepted message. Thus, sensor u discards the message.

Note that sensor u cannot receive a new message while u is in a forwarding period. This is because sensor 0 can broadcast a new message only after the flooding of the previous message is done.

A formal specification for sensors $1..n-1$ is as follows.

```
sensor 1..n-1

const hmax    : integer, {max hop count}
      tmax    : integer, {max waiting time}
      d       : integer  {deafness period}
var   st      : 0..2,    {state, init. 0}
      h,hlast : 0..hmax, {rcvd,last hop cnt}
      s,slast : 0..1,    {rcvd,last seq num}
      t       : 1..tmax  {forwarding time}
begin
   timeout-expires ->
              if st!=1 -> st := 0
              [] st=1  ->
                 send data(hlast,slast);
                 st := 2;
                 timeout-after d+1
              fi

[] rcv data(h,s) ->
      if st=0 or (st>0 and s!=slast) ->
          {accept msg}
          slast := s;
          if h>0 -> st := 1;
                    hlast := h-1;
                    t := random;
                    timeout-after t
          [] h=0 -> st := 2;
                    timeout-after d+1
          fi
      [] st>0 and s=slast -> skip
      fi
end
```

Theorem 3 *(The Multi-Flood Period Theorem)*

$$m \geq hmax * tmax + 1$$

Proof: When sensor 0 broadcasts a first message at time s, sensor 0 needs to wait until this message is no longer forwarded in the network, before broadcasting a second message with the toggled sequence number. The flood of the first message is guaranteed to be done at $s + hmax * tmax$. Therefore, the earliest time sensor 0 can broadcast the second message is $s + hmax * tmax$+1. The multi-flood period needs to be at least $hmax * tmax + 1$ to guarantee that no forwarded messages from the two floods collide with one another, and every sensor is ready to accept the second message, when sensor 0 broadcasts it. □

To analyze the protocol, we use the minimum possible values for the multi-flood period m and the deafness period d, from Theorem 3 and 1, as follows:

$$d = hmax * tmax$$
$$m = hmax * tmax + 1$$

Adopting the above values of d and m, it is straightforward to show that th mult-flood protocol satisfies the following three properties discussed in Section 1: (i) Few collisions within a single flood. (ii) No collisions of consecutive floods. (iii) No redundant forwarding.

Note that in this protocol, when a sensor receives a message with sequence number 0, if the sensor is ready to accept a message with sequence number 0 (i.e. either in an accepting state, or in a deafness state with the sequence number of the last accepted message equal to 1), then the message is new and the sensor accepts it. Moreover, sensor 0 broadcasts a new message with sequence number 0 only when every sensor is ready to accept a message with sequence number 0. Therefore, the sensor does not discard any new message with sequence number 0, if the sensor receives it. Similarly, the sensor does not discard any new message with sequence number 1, if the sensor receives it.

Table 2 shows the multi-flood period over various $tmax$ and $hmax$ values. The multi-flood period is reduced to the half of the flood period of the disciplined flood protocol. Note that the reach of the multi-flood protocol is the same as that of the disciplined flood protocol, since sensor 0 initiates the flooding of a next message, after the previous message is no longer forwarded in the network as in the disciplined flood protocol.

tmax	hmax=8	hmax=9	hmax=10
5	41	46	51
10	81	91	101

Table 2. The multi-flood period (in time units)

As similar to the disciplined flood protocol, in practical setting, one may use the half of the multi-flood period, without significant performance degrading of the protocol. However, in this case, the protocol cannot guarantee that no collisions of two consecutive floods happen, and a sensor may fail to distinguish new messages from redundantly forwarded messages.

7. Related Work

Several flood protocols have been proposed to reduce redundantly forwarded messages in a flood [4], [5], [6], [7]. In [3], various flood protocols were categorized based on how a sensor decides whether it forwards a received message or not. Probability, location information, or neighbor information can be used to make this decision. Sun et al [5] also investigate how the waiting time before forwarding a message affects on the performance of flooding. Unlike these protocols, our disciplined flood protocols control the activities within a single flood or across consecutive floods to reduce or prevent message collisions.

Gouda[1] developed the Abstract Protocol (AP) notation to specify and verify network protocols in a high level, and McGuire[2] developed the Timed Abstract Protocol notation based on AP notation, adding the ability to express the temporal behavior. Our notation used to describe the disciplined flood protocols is also based on AP notation, but this notation is to specify and verify the activities of sensor network protocols. In [9], a formal specification, similar to TLA which is used for modeling concurrent system, is used to specify a flood protocol. Using the formal specification of the protocol, the author identifies the conditions that make flooding unreliable and finally proves that flooding is unreliable. Based on this formal model, Downey *et al* develop a simulator and evaluate the performance of flooding. This simulator can adopt various models for radio, transmission, media access control, etc, to achieve accuracy or efficiency of the simulation.

Many simulation frameworks have been developed for sensor networks such as TOSSIM[12], SensorSim[13], Prowler[14], EmStar[17], VisualSense[15]. In general, these simulation frameworks focus on the accurate and detailed simulation of sensor network protocols, to evaluate the performance of the protocol implementations. They also provide mechanisms to adopt various components or models to their frameworks. However, we developed our simulator to evaluate the performance of the protocol design itself and to verify the behavior of a protocol specified in our formal model.

EnvioTrack[11] provides a high-level programming abstraction for sensor network applications, specially tracking applications. Using this abstraction, a programmer can easily implement a sensor network application, without developing lower level components such as group management and routing. In [16], Volgyesi *et al* introduce an interface modeling language to describe interface specifications of components. This language can be used to verify the design and composition of components.

Ganesan *et al* [8] study the performance of a flood protocol based on experiment over 150 motes. They evaluate the effect of each network layer such as physical and link, medium access, and network and application layers on the performance of the flood protocol.

8. Concluding Remarks

Flood is a communication primitive that can be initiated by the base station of a sensor network to send a copy of some message to every sensor in the network. When a flood of some message is initiated, the message is forwarded by every sensor that receives the message until the sensors decide not to forward the message any more. This uncontroled flood can cause the following three problems: (i) Collisions within a single flood. (ii) Collisions of consecutive floods. (iii) Redundant forwarding.

We presented a family of disciplined flood protocols, that aim to prevent or reduce the above three problems. To reduce collisions within a single flood, when a sensor receives a flood message, it waits for a random forwarding period, before sending the received message. To prevent collisions of consecutive floods, sensor 0 needs to wait for the flood period after sensor 0 broadcasts a message and before it broadcasts a next message. At last, to prevent redundant forwarding, once a sensor accepts and forwards a flood message, the sensor restrains from accepting or forwarding all received messages for the deafness period.

The simulation result showed that while a regular flood protocol can cause a flooded message to reach between 60% and 80% of all sensors in the network, our disciplined flood protocol can cause a flooded message to reach between 88% and 99% of all sensors in the network.

In the disciplined protocol, sensor 0 may need to wait for a long time before broadcasting a next message to achieve a high reach of flood. To increase the frequency of the protocol, we developed the multi-flood protocol that attaches one bit sequence number to each flood message and uses it to distinguish new messages from redundantly forwarded messages. This multi-flood protocol can reduce the flood period by a factor of two.

In our flood protocols, once a sensor receives a flood message, the sensor does nothing but is waiting for a forwarding period to finish or for a deafness period to finish. Thus, the sensor can go to sleep during these periods to save its energy.

When the disciplined flood protocol uses large values for $tmax$ and $hmax$ to achieve a high reach of flood, the latency of the protocol becomes high. One way to reduce the latency while having a high reach is as follows: When a sensor has a message to forward, the sensor sends the message more than one time. In this way, the protocol can achieve a high reach of flood even when the value of $tmax$ is small, resulting in the low latency of the protocol.

Acknowledgment

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Contract F33615-01-C-1901. The authors would like to thank Dr. Mehmet Karaata for interesting discussions.

References

[1] M. G. Gouda, *Elements of Network Protocol Design*, John Wiley and Sons, Inc, New York, New York, 1998.

[2] T. M. McGuire. Correct Implementation of Network Protocols. Ph. D. Dissertation, The University of Texas at Austin, 2004.

[3] B. Williams and T. Camp. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing* (MOBIHOC), 2002, pp. 194–205.

[4] S. Ni, Y. Tseng, Y, Chen, and j. Sheu. The Broadcast Strom Problem in a Mobile Ad Hoc Network. In Proceeding of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pages 151-162, 1999.

[5] M. Sun, W. Feng, and T. Lai. Location Aided Broadcast in Wireless Ad Hoc Networks. In Proceeding of the IEEE GLOBECOM 2001, pp. 2842-2846, November 2001.

[6] H. Lim and C. Kim Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks. In *Proceedings of the ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWIM), 2000.

[7] R. Gandhi, S. Parthasarathy and A. Mishra. Minimizing broadcast latency and redundancy in ad hoc networks. In Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing, Maryland, 2003.

[8] 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, Mar 2002.

[9] R Cardell-Oliver. Why Flooding is Unreliable in Multi-hop, Wireless Networks. February 2004, Submitted for Review.

[10] P. Downey and R. Cardell-Oliver. Evaluating the Impact of Limited Resource on Performance of Flooding in Wireless Sensor Networks. In Proceeding of the International Conference on Dependable Systems and Networks, Florence, July, 2004.

[11] T. Abdelzaher, B. Blum B, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru and A. Wood. EnvioTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In Proceeding of the 24th International Conference on Distributed Computing Systems. Tokyo, Japan. March 23-26, 2004.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)

[13] S. Park, A. Savvides and M. B. Srivastava. SensorSim: A Simulation Framework for Sensor Networks. In the Proceedings of MSWiM 2000, Boston, MA, August 11, 2000.

[14] G. Simon, P. Volgyesi, M. Maroti, A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In Proceedings of the IEEE Aerospace Conference, March 2003.

[15] P. Baldwin, S. Kohli, E. Lee, X. Liu, Y. Zhao. Modeling of Sensor Nets in Ptolemy II. In Proceedings of Information Processing in Sensor Networks (IPSN), April, 2004, Berkely, CA.

[16] P. Volgyesi, M. Maroti, S. Dora, E. Osses, A. Ledeczi, T. Paka. Embedded Software Composition and Verification Technical Report ISIS-04-503, Vanderbilt University, 2004

[17] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, D. Estrin. EmStar: a Software Environment for Developing and Deploying Wireless Sensor Networks. In the Proceedings of USENIX General Track 2004.