A Multi Sensor Flood Sequencing protocols in Sensor Networks

Pooja H[1], Anitha T N[2]

Department of Computer Science & Engineering,
S.J.C. Institute of Technology, Chickballapura, Karnataka.

[1]pooja.h.28@gmail.com
[2]nisureddy@rediffmail.com

***Abstract*— Wireless sensor networks are potentially one of the most important technologies of this century. Recent advancement in wireless communications and electronics have enabled the development of low-cost, low-power, multifunctional miniature devices for use in remote sensing applications 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. 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.**

A stabilization properties of these four flood sequencing protocols will be analyzed for multiple sources. The objective is that the differentiated sequencing protocol has better stabilization property and provides better performance than those of the other three protocols.

***Keywords*— Flood sequencing protocols, multiple sources, sequence numbers, sensor networks, Redundant and fresh message.**

I. INTRODUCTION

Wireless sensor networks are potentially one of the most important technologies of this century. Recent advancement in wireless communications and electronics has enabled the development of low-cost, low-power, multifunctional miniature devices for use in remote sensing applications. The combination of these factors has improved the viability of utilizing a sensor network consisting of a large number of intelligent sensors, enabling the collection, processing analysis and dissemination of valuable information gathered in a variety of environments. A sensor network is composed of a large number of sensor nodes which consist of sensing, data processing and communication capabilities. These nodes are densely deployed either inside the phenomenon or very close to it.

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 neighbours. 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 neighbours. 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. 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. 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.

II. SYSTEM ANALYSIS

A. Analysis on Existing Networks

The practice of using sequence numbers to distinguish between fresh and redundant flood messages has been 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 information or neighbour and history information of the flood protocols discussed in [3],[4],[5],[6],[13],[15] assuming 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.

It was suggested to associate a sequence number with each flood message. The flood protocols discussed in [7], [8], and [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. 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.

B. Idea on Proposed Network

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.

For multiple sources, the flood protocols 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 to distinguish fro multiple sources. Similarly, it was suggested in that each node maintains a list of flood messages received by the node recently. , for a recently received flood message, each node maintains an entry of the source address, sequence number, and lifetime.

III. SENSOR NETWORK MODEL

A. Network Characteristics and Design Objectives

The characteristics of sensor networks and application requirements have a decisive impact on the network design objectives in term of network capabilities and network performance.

B. Network Characteristics

As compared to the traditional wireless communication networks such as mobile ad hoc network (MANET) and cellular systems, wireless sensor networks have the following unique characteristics and constraints:

1. *Dense sensor node deployment:* Sensor nodes are usually densely deployed and can be several orders of magnitude higher than that in a MANET.
2. *Battery-powered sensor nodes:* Sensor nodes are usually powered by battery and are deployed in a harsh environment where it is very difficult to change or recharge the batteries.
3. *Severe energy, computation, and storage constraints:* Sensors nodes are having highly limited energy, computation, and storage capabilities.
4. *Self-configurable*: Sensor nodes are usually randomly deployed and autonomously configure themselves into a communication network.
5. *Unreliable sensor nodes:* Since sensor nodes are prone to physical damages or failures due to its deployment in harsh or hostile environment.
6. *Data redundancy:* In most sensor network application, sensor nodes are densely deployed in a region of interest and collaborate to accomplish a common sensing task. Thus, the data sensed by multiple sensor nodes typically have a certain level of correlation or redundancy.
7. *Application specific:* A sensor network is usually designed and deployed for a specific application. The design requirements of a sensor network change with its application.

8. *Many-to-one traffic pattern*: In most sensor network applications, the data sensed by sensor nodes flow from multiple source sensor nodes to a particular sink, exhibiting a many-to-one traffic pattern.
9. *Frequent topology change:* Network topology changes frequently due to the node failures, damage, addition, energy depletion, or channel fading.

C. Network Design Objectives

Most sensor networks are application specific and have different application requirements. Thus, all or part of the following main design objectives is considered in the design of sensor networks:

1. *Small node size*: Since sensor nodes are usually deployed in a harsh or hostile environment in large numbers, reducing node size can facilitate node deployment. It will also reduce the power consumption and cost of sensor nodes.
2. *Low node cost:* Since sensor nodes are usually deployed in a harsh or hostile environment in large numbers and cannot be reused, reducing cost of sensor nodes is important and will result into the cost reduction of whole network.
3. *Low power consumption:* Since sensor nodes are powered by battery and it is often very difficult or even impossible to charge or recharge their batteries, it is crucial to reduce the power consumption of sensor nodes so that the lifetime of the sensor nodes, as well as the whole network is prolonged.
4. *Scalability*: Since the number sensor nodes in sensor networks are in the order of tens, hundreds, or thousands, network protocols designed for sensor networks should be scalable to different network sizes.
5. *Reliability:* Network protocols designed for sensor networks must provide error control and correction mechanisms to ensure reliable data delivery over noisy, error-prone, and time-varying wireless channels.
6. *Self-configurability:* In sensor networks, once deployed, sensor nodes should be able to autonomously organize themselves into a communication network and reconfigure their connectivity in the event of topology changes and node failures.
7. *Adaptability*: In sensor networks, a node may fail, join, or move, which would result in changes in node density and network topology. Thus, network protocols designed for sensor networks should be adaptive to such density and topology changes.
8. *Channel utilization:* Since sensor networks have limited bandwidth resources, communication protocols designed for sensor networks should efficiently make use of the bandwidth to improve channel utilization.
9. *Fault tolerance:* Sensor nodes are prone to failures due to harsh deployment environments and unattended operations. Thus, sensor nodes should be fault tolerant and have the abilities of self testing, self-calibrating, self-repairing, and self-recovering.
10. *Security:* A sensor network should introduce effective security mechanisms to prevent the data information in the network or a sensor node from unauthorized access or malicious attacks.
11. *QoS support:* In sensor networks, different applications may have different quality-of-service (QoS) requirements in terms of delivery latency and packet loss. Thus, network protocol design should consider the QoS requirements of specific applications.

D. Sensor Network Execution

Assuming 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. Refering to the time period between two consecutive instants t and t þ 1 as a time unit (t; t þ 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 þ 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 þ 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 timeout action, the value of timer.u in the time unit (t; t þ 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-neighbour 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 neighbour of v at t. If the message sent by u collides with another message, then v receives no message at t. 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.

IV. Overview Of Flood Sequencing Protocols From Multiple Sources

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(w,hmax, slast), where w is the source ID, hmax is the maximum number of hops to be made by this data message in the network,slast is the last sequence number.

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

```
sensor   u.: 1….n-1
const   hmax : integer          { max hop count }
tmax : integer       { max forwarding period }
var   h,hlast : 1…hmax   {rcvd,last hop count }
s,slast : integer { rcvd,last seq nmber}
new : Boolean      {true if u has msg to forward}
begin
timeout-expires -> if new -> new= false
                                send data (w, hlast, slast);
                                [] new -> skip
                                fi,
timeout-after random(l,tmax)

[] rcv data(w,h,s) -> if { the mgs is fresh}
                      {accept msg} slast = s;
                      if h>1 ->new =true
                      hlast=h-1
[] h<=1  ->skip
fi
[] {the msg is redundant} - >{discard msg} skip
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.

A. First Protocol: Sequencing Free

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 timeout action of sensor 0 is specified as follows:

```
timeout-expires -> { generate new msg}
send  data(w,hmax, s);
timeout-after f
```

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 w0, 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 w0. 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 onverges to a legitimate state, provided that each source keeps initiating floods and the assumption of bounded message loss in is satisfied for each source.

```
timeout-expired ->if new -> new=false;
                  send data(w,hlast)
                  []->new ->skip
        fi,timeout-after random(1,tmax)
[] rcv data(w,h,s) -> {accept msg}
        if h>1 ^ → new ->new =true
                  hlast=h-1
        [] h<=1 v new ->skip
        fi
end
```

B. Second Protocol: Linear Sequencing

In this section, 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 u broadcasts a new message, sensor u 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:

```
timeout-expires -> { generate new msg}
slast = slast -1;
send  data(w,hmax,slast);
timeout-after f
```

When sensor u receives a data(w, 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:

```
[] rcv data(w,h,s) -> if s>slast -> {accept msg }
                  slast =s;
                  if h>1 →new =true
                  hlast=h-1
[] h<=1 ->skip
fi
[] s<=slast -> {discard msg} skip
fi
end
```

C. Third Protocol: Circular Sequencing

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 u broadcasts a new message, sensor u 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 u is modified as follows:

```
timeout-expires -> {generate new msg}
slast = slast -1 mod (smax +1);
send  data(w,hmax,slast);
timeout-after f
```

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:

```
[] rcv data (w, h, s) -> if Larger(s, slast) -> {accept msg}
                         slast =s;
                         if h>1 →new =true
                         hlast=h-1
        [] h<=1 ->skip
        fi
        [] Larger(s, slast) -> {discard msg} skip
  fi
```

V. Fourth Protocol: Differentiated Sequencing

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 smax > 0.We assume that smax is an even Sensor u in this protocol is identical to the one in the circular sequencing protocol. However, when a sensor u receives a data (w, 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:

```
[] rcv data(w,h,s) -> if  s != slast  -> {accept msg }
                      slast =s;
                      if h>1 →new =true
                      hlast=h-1
                      [] h<=1 ->skip
                      fi
       [] s= slast -> {discard msg} skip
       fi
```

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 w0, 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 w0. 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 is satisfied for each source.

VI. Conclusion And Future Plan

A family of the four flood sequencing protocols that use sequence numbers and source sensors 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. Conclusions made that the differentiated sequencing protocol has better overall performance in terms of communication, and stabilization and stable properties. The 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 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.

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. Heissenbu¨ ttel, T. Braun, M. Waelchli, and T. Bernoulli, "*Optimized Stateless Broadcasting in Wireless Multi-Hop Networks*," Proc. IEEE INFOCOM, 2006.