ENERGY-EFFICIENT SHORTEST PATH SELF-STABILIZING MULTICAST PROTOCOL FOR MOBILE AD HOC NETWORKS

by

Ganesh Sridharan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

May 2004

ENERGY-EFFICIENT SHORTEST PATH SELF-STABILIZING MULTICAST PROTOCOL

FOR MOBILE AD HOC NETWORKS

by

Ganesh Sridharan

has been approved

May 2004

APPROVED:

_____, Chair

_____

_____

Supervisory Committee

ACCEPTED:

_____

Department Chair

_____

Dean, Graduate College

ABSTRACT

Multicasting in mobile ad hoc networks (MANETs) has gained attention with the growth of research on ad hoc networks. Self-stabilizing multicast protocols for MANETs are an entirely new set of protocols that apply the concept of self-stabilization to construct multicast trees. Self-stabilizing distributed algorithms converge to a global legitimate state in presence of any intermittent faults and still use only local knowledge for actions at each node and hence, they are fault-tolerant. Further, these algorithms build multicast trees in a *proactive* fashion. Proactive schemes are attractive because they consume fixed control overhead and have lesser delay compared to on-demand schemes. In this work, we analyze the performance of Self-Stabilizing Shortest Path Spanning Tree (SS-SPST) protocol for MANETs through simulation and implementation and compare with that of other MANET multicast protocols. Further, we propose an energy-efficient cost metric to SS-SPST protocol. This cost metric reflects not only on the transmission power required to transmit but also on the energy spent in discarding packets by nodes that are not involved in the multicast operation. Based on simulation results, we discuss the trade-offs involved in saving energy. Further, we implement the protocols in a real MANET and analyze the results for correctness.

To my parents

ACKNOWLEDGMENTS

I gratefully acknowledge the guidance of my advisor, Dr. Sandeep Gupta, from whose proposal this thesis grew. I am indebted to members of my committee, Dr. Arunabha Sen and Dr. Goran Konjevod for insightful discussions and helpful advice. I thank Georgios Varsamopoulos for his detailed comments and suggestions.

I also thank NSF grant - ANI-0196156 for funding this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Mobile ad hoc networks (MANETs) [18] are formed in an ad hoc fashion without any underlying wired infrastructure. A MANET consists of a dynamic collection of mobile nodes, which have limited wireless bandwidth to transmit/receive messages. Since the transmission range of each node is limited, messages from one node might not reach all the intended receivers. Due to this limited transmission power, routes in a MANET are often multi hop. Nodes in these networks move in a random fashion and their battery power is also limited. MANETs find use in applications such as military operations and law enforcement. Most of these applications require one to many communications. Multicasting is the transmission of datagrams to a group of hosts identified by an unique address [17]. Designing a multicast protocol for a dynamically re-configurable network such as a MANET is a challenging problem. One possible solution to multicast can be the use of *flooding* approach. However, this solution suffers from redundant rebroadcast, collision and contention as mentioned in [12]. We believe that constructing a structure such as a tree or a mesh to deliver multicast messages can overcome these shortcomings.

Mobile ad hoc networks complicate the multicasting operation since the underlying topology changes very frequently. Whenever a node moves out of the multicast structure, the protocol has to rebuild the structure to include the node. The node movement, and hence the breakage in the multicast structure, can be seen as a fault occurring in the system and the protocol as an agent attempting

to rectify the fault. In many of the existing multicast protocols for the MANETs, the responsibility to correct the fault is given to a single node. It would have been indeed good to design a system that corrects the fault by itself without any external agents. In other words a fault-tolerant system will perform better compared to a system which masks the faults. Self-stabilizing system provides an answer to design a fault-tolerant system by constructing tree structure to multicast messages in MANETs.

Dijkstra proposed the idea of self-stabilization [13], [14]. Self-stabilizing distributed algorithms converge to a global legitimate state in presence of any number of intermittent faults while using only local knowledge for actions at each node. Hence, self-stabilizing protocols provide means for tolerating transient faults. However, self-stabilizing protocols provide a *non-masking approach* to fault tolerance. The service provided by a self-stabilizing protocol may be unavailable or partially available while the protocol is stabilizing. Further, the self-stabilizing protocols only guarantee that the protocols will *eventually* stabilize. Hence, they provide no guarantee on how soon the service will become available after the occurrence of faults. Thus, these genre of protocols provide a best-effort service in a faulty environment without any guarantees. SS-SPST [1] is a distributed, self-stabilizing algorithms to maintain the multicast tree in a given MANET for a given set of nodes. This algorithm brings practical use to the self-stabilization paradigm by applying it to build multicast trees in MANETs. This algorithm is fault-tolerant since it can detect link failures due to node movements and readjust the multicast tree.

[4] gives a detailed list of the energy concerns in wireless networks. In this work we consider mobile ad hoc nodes with battery power as the source of energy, which means that the source of energy is limited and non-renewable. Since energy in these nodes is limited, each node has to use it effectively to avoid early termination of the network. Multicasting in mobile ad hoc networks involves“relaying” of messages from one group node to other group node using forwarding nodes.

Further, the broadcast nature of the wireless medium makes non-group nodes overhear and discard data packets destined for multicast group nodes. By energy efficiency we mean to effectively use available energy in transmitting, receiving, forwarding and discarding packets.

As proposed, SS-SPST uses hop count as the cost metric to construct the shortest path spanning tree. This metric does not reflect the true nature of the wireless networks. Successful reception at the receiving node depends on noise, interference, channel fading, bandwidth etc.. The presence and absence of a link is determined by the transmission power of the sender and the energy level at the sender and receiver nodes. Thus the cost metric for a multicast operation needs to take few of these factors, if not all, into account.

Multicasting operation in a MANET involves significant energy spent by the participating nodes. In a MANET, the multicast operation is ultimately limited by the constraint of finite battery life at the individual users. This constraint is important in military operations and other life-critical operations because it may be impossible to recharge batteries during the course of a mission or emergency situation. Thus, there is a need to develop multicast protocols which make efficient use of limited energy available.

In this work, firstly, we simulate SS-SPST with hop count metric and analyze its multicast performance and compare it with that of other MANET multicast protocols like Multicast Operation of the Ad Hoc On-Demand Distance Vector Routing Protocol (MAODV) [10] and On-Demand Multicast Routing Protocol (ODMRP) [11]. Secondly, we propose an energy-efficient cost metric for SS-SPST algorithm. This cost metric reflects the maximum transmission energy required to reach the farthest neighboring node and the energy expended by all other neighboring nodes which are not part of the multicast communication. We discuss the trade-offs involved in saving energy and support our claims with simulation results. For convenience, we name the SS-SPST algorithm with our cost metric as SS-SPST-E. Finally, we implement the protocols SS-SPST and SS-SPST-E

in a real mobile ad hoc environment using IEEE 802.11 wireless LAN (Local Area Network) cards and analyze the results for correctness.

CHAPTER 2

Motivation

1. Need for a new MANET multicast protocol

Unlike other popular MANET multicast protocols like MAODV [10] and ODMRP [11], SS-SPST is a multicast protocol which maintains a multicast tree pro actively. We believe, proactive multicast protocols are suited when delay and throughput are main considerations. For example, consider the following situation. Joe handles a class room of students with different majors. Each student has his/her own personal computing device with wireless access and uses SS-SPST. If Joe wants to show a video clip for students who major in chemistry, he can simply multicast the video clip to the chemistry group. This is a typical scenario where we need less delay and high packet delivery ratio. Self-stabilizing protocols will perform well compared to other on-demand multicasting protocols for mobile ad hoc networks in these kind of scenarios. Pro activeness comes with a cost. We need to maintain the multicast tree even when there are no multicast messages. Thus SS-SPST might have more control overhead when there are no multicast messages compared to that of on-demand versions. Thus it becomes imperative to evaluate the performance of SS-SPST and compare with other MANET multicast protocols to establish the usefulness of SS-SPST.

2. Need for a Dynamic Cost Metric

SPST constructs the multicast tree by minimizing the cost needed to reach the source node from the destination. Currently it assumes unit cost over the links. Cost of a link in the tree reflects the price to be paid to transmit a packet over that link. Traditionally, many algorithms that are based on Dijkstra's shortest path algorithm, assume links with fixed cost and try to minimize the hop count to the destination. This assumption, to some extent, captures the real picture in wire line networks but in wireless networks, this assumption fails miserably.

To illustrate that minimizing distance always will not be as efficient in wireless networks as in wire line networks, we consider the following example. In figure 1, we consider a source node S and two destination nodes D_1 and D_2. Let the distance between S and D_1 be a, the distance between S and D_2 be c and the distance between D_1 and D_2 be b. Let us assume without loss of generality that $a + b > c > a$ and θ be the angle $\angle D_1 S D_2$. We know that power needed to support a link in a wireless network is proportional to square of the distance in the simplest case. So

$P_1 = a^2$ is the power required to support the link between S and D_1

$P_2 = c^2$ is the power required to support the link between S and D_2

$P_{12} = b^2$ is the power required to support the link between $D1$ and D_2

Now node S can reach node D_2 either directly by transmitting with power P_2 or through node D_1 with total power being $P_1 + P_{12}$. Minimum distance algorithm will always choose the first option and directly transmit to node D_2 because $a + b > c$. When $a < c\, cos\theta$ then the option of transmitting through D_1 is energy effective since $a^2 + b^2 < c^2$. Thus we see that the option of using minimum distance alone might not be energy efficient all the time. In this example the energy value is determined by the value of θ which changes with time. Hence, we cannot use just distance as the

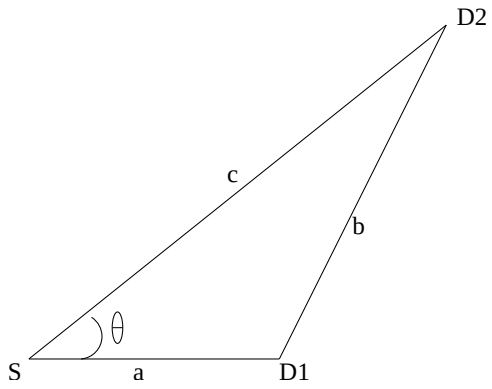

Figure 1. Wireless Routing

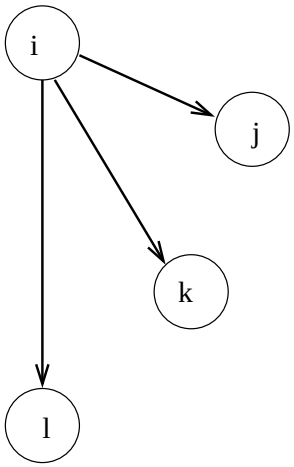

Figure 2. Wireless Multicast Advantage

metric and we need a dynamic cost metric to capture the changes in the wireless environment.

There are many factors that influence the links in wireless networks like link error rate, node energy level etc., which are time variant. According to the nature of the cost factor we can broadly classify the cost either as a *Link Cost* or as a *Node Cost*. Link costs capture the link variants such as error rate, channel bandwidth and transmission power while, Node costs capture the node variants such as energy level and node load.

3. Wireless Multicast Advantage

Wieselthier in [6] has mentioned about the advantages of multicast in wireless networks. When developing a multicast protocol for MANETs, one can exploit the broadcast property of wireless communication. Consider an example in2, in which a subset of node involves a source node i transmitting to its neighbor nodes j, k and l. Let us further assume that node l is the farthest neighbor from the source node i. When node i wants to transmit a packet to all the neighboring nodes, unlike wired networks, it can just send a single packet instead of sending three individual unicast packets. All the three neighbors will receive the packet because of the broadcast nature of the wireless networks. When node i multicasts a packet, it multicasts with a transmission range required to reach the farthest neighboring node, l and all the nodes within this transmission range receive a packet. This advantage of sending a single broadcast packet instead of sending individual unicast packets is termed as *wireless multicast advantage*.

Though the broadcast property of the wireless networks works as an advantage, there are few inherent disadvantages too. In the example given in figure 2 if node j is not a part of the multicast group, it has to receive the packets transmitted by source node i and discard it. We term the energy expended in discarding packets as discard energy. One of the main goals of this work is to minimize the total discard energy expended by the wireless system.

CHAPTER 3

System Model

To implement SS-SPST and SS-SPST-E protocols in a distributed manner, we assume the presence of a self-stabilizing system with few properties. In this section, we discuss the properties of self-stabilizing systems, assumptions made and energy model adopted in our system.

1. Definition of self-stabilization

A distributed system is modeled to be composed of two types of *components*, processors and interconnection between processors. A processor is a computer machine which executes a program and these processors can be interconnected with neighboring processors by various interprocess communication techniques like message passing or shared memory. *Topology* of a distributed system can be represented as a graph containing processors as vertexes and interconnections between processors as edges. Each component in the distributed system has a *local* state and the *global* state of the system is the union of the local states of all the components. A predicate P, defined over the set of global state, represents the correct execution condition.

A distributed system S is self-stabilizing with respect to the predicate P, if it satisfies the following two conditions:

1. **Convergence**: Starting from any arbitrary global state, S is guaranteed to reach a global state,

satisfying P in finite number of state transitions.

2. **Closure**: Once S reaches a global state satisfying P, it cannot be falsified.

SS-SPST protocol uses self-stabilization technique to build multicast tree in a mobile ad hoc environment. It uses $beacon$ messages to achieve intercommunication between nodes in a wireless system. Beacon message is a message which is broadcast periodically by each node within its coverage area to advertise its presence to other neighboring nodes. It contains local information about the sender node and is used by the receiver node to update changes in the network topology.

The usual time-complexity measure for self-stabilizing algorithms is that of *rounds* [20]. In asynchronous systems, a round ends in a computation when every processor has executed at least one step. In a wireless system such as a MANET, communication messages are bound to be lost because of the lossy nature of the wireless medium. In such a lossy system,

Definition 1. *A round is defined to be the time period in which each node in the system receives at least one beacon message from each of its neighbors and performs computation based on the information it has received.*

2. Assumptions

Implementation of SS-SPST in a wireless network is based on the following assumptions.

- **Unique identification and neighboring:** Each node in the mobile ad hoc network is identified by a unique identifier. When a node transmits (*broadcasts*) a message, the nodes in the *coverage area* of the sender can (almost) simultaneously hear the message. A node i is called

a *neighbor* of node j iff it is in node j's coverage area. This relationship is time varying since the nodes are mobile. At any time a node i can correctly receive a message from one of its neighbors, say j, iff j is the only neighbor of i which is transmitting at that time. A medium access protocol is used for collision-free access of the shared wireless communication medium.

- **Beaconing:** Each node periodically broadcasts a $beacon$ message. The nodes in the network identify their single hop neighbors using this message. Each node maintains a list of neighbors and timers for its neighbor link. When a node i receives a beacon message from a node j, which is not in its neighbor list, it adds the new node j to its list of neighbors and starts a timer for that link t_{ij}. If node j is already a neighbor then node i resets the timer t_{ij}.

- **Soft-state neighboring:** When the timer t_{ij} expires, node i removes node j from its neighbor list.

- **Cost metric computation:** When a node j sends a beacon message to any of its neighbors, say node i, it includes node and link characteristic information that is used by node i to calculate the cost of link (i, j) and other values to construct the tree.

- **Variable transmission range:** A node can dynamically vary its transmission range without spending any additional energy. A node transmits beacon messages with the maximum transmission power and data messages with a transmission power calculated based on the distance of the farthest group neighbor. Further, all messages are sent as broadcast messages to exploit the *wireless multicast advantage* property [6].

- **Active mode:** All the nodes operate in *active* mode and hence listen to all broadcast messages. They act upon the multicast data message based on their multicast group information. Thus,

all non-group nodes in the transmission range of a particular node will expend energy to listen to the data message and discard it.

- **Topology model:** The topology of the ad hoc network is modeled by an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of links between neighboring nodes. Since the nodes are mobile, the network topology changes with time. All the links in the graph are considered to be bi-directional and hence, graph G is a symmetric graph.

- **Multicasting:** We assume a single-source multicasting with root of the tree as the source. Multicasting data messages are sent by the source as downstream stream messages to all the group nodes in the tree. Thus, multicast data messages traverse only through downstream links of the tree.

3. Energy Model

We adopt an energy model proposed in [21]. This model is a first order radio model in which the transmitter consists of transmitter circuitry and an amplifier and the receiver consists of receiver circuitry. The transmitter and receiver circuitry dissipates $E_{\mathsf{elec}} = 50nJ/bit$ to run and the transmitter amplifier dissipates $\epsilon_{\mathsf{amp}} = 100pJ/bit/m^2$ to achieve an acceptable signal to noise ratio. This model assumes an r^2 energy loss due to channel transmission. Thus, to transmit a k-bit message a distance d using this model, the radio expends:

$$E_{\mathsf{Tx}}(k, d) = E_{\mathsf{Tx-elec}}(k) + E_{\mathsf{Tx-amp}}(k, d)$$
$$E_{\mathsf{Tx}}(k, d) = E_{\mathsf{elec}} \cdot k + \epsilon_{\mathsf{amp}} \cdot k \cdot d^2$$

and to receive this message, the radio expends:

$$E_{\mathsf{Rx}}(k) = E_{\mathsf{Rx-elec}}(k)$$

$$E_{\mathsf{Rx}}(k) = E_{\mathsf{elec}} \cdot k$$

Further, this model assumes that the radio channel is symmetric such that the energy required to transmit a message in one direction is the same as that of in the other direction.

CHAPTER 4

Multicast Protocols in Mobile Ad Hoc Networks

1. Shortest Path Self-Stabilizing Protocol

This section gives a brief overview on SS-SPST, proposed by Gupta et al. in [1]. We describe the shortest path spanning tree algorithm, the multicast protocol built on it and some important results. For completeness sake, we have listed a few definitions and results given by Gupta et al. in [1] and [2]. For a detailed set of proofs refer to Idem. [1] and [2].

SS-SPST constructs a multicast tree in a MANET in three logical steps. Firstly, it constructs a shortest-path spanning tree of the mobile network graph in presence of topology change due to node mobility. Secondly, the algorithm roots the source node with respect to the stabilized multicast tree. Finally, it prunes the nodes from the constructed tree that are not involved in multicasting.

Let $G = (V, E)$, a symmetric graph with no self-loops, represent the mobile ad hoc network where V is the set of nodes (mobile hosts), $|V|$ = n and E is the set of edges/links. Let m_1 and m_2 denote respectively the minimum and maximum edge cost in the system graph and let $\mathcal{D}$ be the diameter of the underlying network in terms of the number of edges traversed. Each node in the network attempts to compute the cost of the shortest path to the given reference node r. In this case, the authors in [1] assume node r to be the source of the multicast session. Let $S_i(r)$ denote the cost

of the shortest path from node i to node r.

Remark 1. *For all i, $S_i(r)$ is determined by the topology of the graph and the costs assigned to the edges and that $S_r(r)$ = 0 since we assume the graph G has no self loops [2].*

1.1. The algorithm. Each node $i \in V$ maintains a local variable $D_i(r)$, current estimate of $S_i(r)$ known at node i. In addition, each node i maintains a variable P_i, to hold the identifier of the parent. Parent identifier is the identifier of a node in the set of neighboring nodes denoted by the set $Adj(i)$; P_i points to the node adjacent to node i in the currently estimated shortest path from node i to node r. For each node i, the values of $D_i(r)$ and P_i determine the local state of node i.

Remark 2. *In an illegitimate state any node i (including the root node r) can have an arbitrary value for $D_i(r)$ between 0 and large positive number which we shall call MAXINT (determined by the length of the registers holding these variables). Similarly,in an illegitimate state, any node can point to an arbitrary node to be its immediate predecessor in the shortest path spanning tree [2].*

Let the set $\mathcal{N}(i)$, for any node i in the graph be defined as,

$$\mathcal{N}(i) = \{j \mid j \in Adj(i) \wedge D_j(r) + w_{ij} = \min_{k \in Adj(i)} (D_k(r) + w_{ik})\}.$$

The set $\mathcal{N}(i)$ contains neighboring nodes of i that are on currently estimated shortest paths from node i to r. Now the legitimate state of the system can be defined as follows

Definition 2. *The system is in the legitimate state iff*

$$\forall i \in V : D_i(r) = S_i(r) \wedge ((i = r \wedge P_i = NULL) \vee (i \neq r \wedge P_i \in \mathcal{N}(i)))$$

Any state which is not legitimate is an illegitimate or unstable state. The purpose of SS-SPST algorithm is to bring the system back to the stable or legitimate state whenever it enters an illegitimate state due to perturbation. Each node looks at the states of its neighbors and checks the local satisfiability of the global legitimate state; if the node is not locally legitimate, it triggers self-stabilization routine and tries to be in the legitimate state. More formally, each node i executes the following code if node i is not locally legitimate.

$$\begin{aligned}
&\text{if } (i = r \wedge (D_i \neq 0 \vee P_i \neq NULL)) \\
&\text{then } D_i = 0 \& P_i = NULL \\
&\text{else if } \Big(i \neq r \wedge (D_i(r) \neq \\
&\qquad \min_{\forall j \in Adj(i)} (D_j(r) + w_{ij}) \vee P_i \notin \mathcal{N}(i)) \Big) \\
&\text{then } D_i(r) = \min_{\forall j \in Adj(i)} (D_j(r) + w_{ij}) \& P_i = k, k \in \mathcal{N}(i)
\end{aligned}$$

We list the lemmas which lead to the proof of correctness of SS-SPST algorithm. For proofs of these lemmas refer to [2].

Lemma 1. *Starting from a given illegitimate state consider the system after p rounds; each of the nodes that are yet to be stabilized has* $D_i(r) \geq pm_1$

Lemma 2. *Consider a node i which is p hops away from root r. Node i will be stabilized in at most* $p\lceil \frac{m_2}{m_1} \rceil$ *rounds after the node* r *is stabilized.*

Lemma 3. *The upper bound on the number of rounds needed by the entire network to stabilize starting from an arbitrary illegitimate state is given by* $\mathcal{D} \cdot \lceil \frac{m_2}{m_1} \rceil + 1$

The knowledge of SS-SPST need not be duplicated at each node; each node will only know its unique predecessor. Each node periodically attempts to calculate the shortest path to the root node using the information given by the neighboring nodes.

Each node in the network keeps information about its multicast group membership and its forwarding node status. Forwarding nodes forward the messages to their children and the multicast group nodes receive the group messages. A multicast group node at any time can terminate its membership in multicast group by just changing its multicast group status. Since this is a self-stabilizing protocol the revoking of one member node is propagated to other nodes eventually. Thus there are no explicit messages required to prune the multicast tree. As nodes in the tree learns about the membership status of its neighbors, it updates its own status.

As we can see, SS-SPST algorithm is based on distance vector routing technique of exchanging local information with neighbors. Most of the algorithms based on distance vector routing suffer from *count-to-infinity* [3] problem. In SS-SPST algorithm, *count-to-infinity* problem is considered as perturbation and the system enters a illegitimate state. SS-SPST algorithm brings back the system to the stable or legitimate state after a finite number of rounds. To state the *count-to-infinity* problem in SPST formally - when the node i, which has a stable path to *root node* r through node l, moves out of the range of node l the link (il) is broken. According to the algorithm, the node $i$tries to estimate the value for $D_i(r)$ and P_i. Assume there is a node j, which is

a neighbor of node i, in the downstream of node i. Node i will choose node j as its parent when $D_j(r) + w_{ij} < D_k(r) + w_{ik}\{k \in Adj(i) \wedge k \neq j\}$. This will lead to *count-to-infinity* problem. Intuitively, we can prove that the system will stabilize, once it encounters the *count-to-infinity* problem. The value of $D_i(r)$ will keep increasing. After some rounds this value will be greater than the value of$D_k(r) + w_{ik}$. Thus the system will go to an illegitimate state and in the worst case the system will take at most $p\lceil\frac{m2}{m1}\rceil$ rounds to stabilize.

Lemma 4. *Count-to-infinity problem in SPST, which directly maps to the illegitimate state in SPST, takes at most* $p\lceil\frac{m2}{m1}\rceil$ *rounds to stabilize*

Proof. When the node i chooses node j as its predecessor node, as explained earlier, $D_i(r)$ is not equal to$S_i(r)$ anymore. Thus the system is not in the legitimate state. Any state other than a legitimate state is an illegitimate state and hence the system will stabilize in $p\lceil\frac{m2}{m1}\rceil$ rounds by Lemma 2. □

2. Multicast Operation of the Ad Hoc On-Demand Distance Vector Routing Protocol - MAODV

This section gives an overview of the MAODV protocol. MAODV multicast protocol is built upon the AODV unicast routing protocol. The multicast routes are formed on-demand using a broadcast route-discovery mechanism. A mobile node sends a Route Request message (RREQ) when it wishes either to send to a multicast group and has no route to that group or to join a multicast group. If a RREQ message is a $join\_request$ then only a member of the multicast group can reply

to that message. If a RREQ message is not a $join\_request$ then any mobile node receiving it and which has a fresh enough route to the group can reply. When an intermediate node which does not have a fresh enough route to a multicast group receives a $join\_request$, it rebroadcasts the request to its neighbors.

A multicast group node which receives a RREQ message with a $join\_request$, sends a unicast RREP reply message to the source. As the reply travels along the path to the source, it is used by the intermediate nodes to update their routing and multicasting table thereby creating a forward path. As the multicast source node receives the RREPs, it keeps track of the route with the latest sequence number and the smallest hop count to the next multicast group member.

After this request-reply phase, the multicast source uses Multicast Activation (MACT) control message to confirm the path from the source to the receiver. It unicasts a MACT message to its selected next hop node and enables the corresponding entry in its multicast route table. The MACT message is forwarded until it reaches the node, which originated the RREP message. The route activation mechanism using MACT messages ensure that the multicast tree structure does not have multiple paths to any tree node.

The first member of the multicast group becomes the leader for that group. The multicast leader sends a GROUP HELLO message periodically containing group IP address and sequence numbers. Nodes use the information in GROUP HELLO messages to update their request table.

At any time a multicast group member may wish to terminate its membership in the multicast group. If the node is not a leaf node in the multicast tree, it may revoke its multicast group member status any time but it should serve as a router for that tree. If the node is a leaf node then it might prune itself from the tree by sending a MACT message with prune flag set. Since the leaf node will have only one node as its tree neighbor, it unicasts this message to that node and the receiving node will prune itself from the tree if its not a multicast group member or the node just

updates its multicast route

3. On-Demand Multicast Routing Protocol - ODMRP

As the name of the protocol suggests, ODMRP is a on-demand multicast routing protocol. It uses *on demand* approach to establish group membership and multicast routes. Similar to MAODV, ODMRP has a $request$ phase and a $reply$ phase. When a multicast source has messages to send, it broadcasts a JOIN QUERY packet to the entire network. This JOIN QUERY packet is periodically broadcast to refresh the membership information, update routes and to overcome the overheads involved in the transmission of $join\_multicast\_group$ and $quit\_multicast\_group$ messages by the multicast receivers.

When an intermediate node receives a non-duplicate JOIN QUERY message, it stores the upstream node ID (i.e. backward learning) and rebroadcasts the packet. When the JOIN QUERY packet reaches a multicast receiver, it creates/updates entries in JOIN TABLE and broadcasts a JOIN REPLY packet. When a intermediate node receives a JOIN REPLY message it checks whether one of the IDs in the next hop list is its own ID. If the ID matches, the node, realizes that its on the path from the multicast source to a multicast receiver, sets is FG-FLAG (Forwarding Group Flag) and updates its own JOIN TABLE based on the matched entries and transmits JOIN REPLY to its next hop node. In this way the JOIN REPLY packet reaches the multicast source node in the shortest path.

The intermediate nodes which forward the JOIN REPLY packets are called forwarding group nodes and they form the mesh structure for multicasting messages. The multicast source, while it has data to send, transmits periodic JOIN QUERY messages to update the structure. Only the non duplicate data packets are forwarded.

As explained earlier, in ODMRP no explicit control packets are needed to be sent by the receivers to join or quit the multicast group. If a multicast source node wants to leave a multicast group, it simply stops sending JOIN QUERY packets to the group and if a receiver wants to leave a multicast group, it stops responding to the JOIN QUERY packets from the source node.

CHAPTER 5

Cost Metrics

SS-SPST constructs the multicast tree by minimizing the cost needed to reach the source node from the destination. The cost of a link in the tree reflects one or more characteristics of the link. Hence, based on the metric that needs to be optimized, we can have different metrics assigned to links. Traditionally, many algorithms that are based on Dijkstra's shortest path algorithm assume links with unit cost and try to minimize the hop count to the destination. This assumption, to some extent, captures the real picture in wire line networks but in wireless networks, this assumption fails miserably. For example, in [4] Ephremides has shown that in ad hoc wireless networks it is energy-efficient to choose long paths along a series of short hops rather than short path along a series of long hops.

There are many factors that influence the links in wireless networks such as link error rate, node energy level etc., which are time variant. According to the nature of the cost factor we can broadly classify the cost either as a *Link Cost* or as a *Node Cost*. Link costs capture the link variants such as error rate, channel bandwidth and transmission power while, Node costs capture the node variants such as energy level and node load. In [4], it has been shown that decreasing the hop count does not necessarily decrease the transmission energy spent. At the same time, increase in hop counts can decrease the performance of protocol since as the number of transmission and reception increases the end-to-end delay increases and the probability of message being lost also increases.

Thus we need to come up with a dynamic cost metric which is both energy-efficient and has minimal impact on the performance of the protocol.

There have been few energy-efficient cost metrics that have been proposed in the literature. As mentioned earlier, most of the metrics can be either classified as node based metric or link based metric. Few link-based metrics have been proposed to achieve energy-efficiency in wireless networks. In [16] and [6] authors have proposed link-based metric which captures the transmission energy spent by the node to maintain a link. By assigning transmission energy expended to the links, SS-SPST builds a tree which minimizes the total transmission energy.

Let T_{ij} be the transmission cost of link between nodes i and neighbor j and let C_{ij} be the total cost of link between i and j. Then we have,

$$C_{ij} = T_{ij}$$

We name the SS-SPST with the above metric to be SS-SPST-T. Where, 'T' denotes the transmission cost. Incorporation of just the transmission energy link metric does not capture the energy expended in real scenarios. Described as the multicast advantage in MANETs, the transmitting node broadcasts the packet to the farthest group neighbor and all other group neighbors within the range receive packet without any extra energy spent as explained before. Thus, SS-SPST-T does not capture the energy spent in a proper way.

Most of the node based metrics, that have been proposed in the literature, take transmission energy or remaining battery level or both into account. For example, in [5] Wieselthier et al. propose schemes(BIP, MIP) to build energy-efficient broadcast and multicast tress in wireless networks. These schemes take the maximum transmission energy of the transmitting node as the node metric. As new nodes join the source node, the transmission energy of the source node keeps increasing.

Applying this method to SS-SPST as a link metric, we can make the farthest neighbor pay the cost of transmission and all the other nodes within the range will have very low overhead cost based on the reception energy.

Let T_{ij} be the transmission cost of link between nodes i and its farthest neighbor j, R be the reception cost and let C_{ij} be the total cost of link between i and j. Then we have,

$$C_{ij} = T_{ij} + R \text{ if } j \text{ is the farthest neighbor}$$

$$C_{ij} = R \text{ otherwise}$$

We name the SS-SPST implementation with the above cost metric as SS-SPST-F for convenience. Where, 'F' signifies the farthest node metric.

Wieselthier et al. in [7] and Chang in [8] proposed a node metric which tries to minimize the total transmission energy and maximize the remaining battery level of each node. In our self-stabilizing system, the faults are caused by changes in topology and incorporation of battery power in the system will lead to additional faults because of frequent changes in battery level. In [9], Banerjee and Misra explain the link-based metric with re-transmission cost. In our protocol, we do not retransmit the lost packets so do not consider the retransmission cost.

We believe that a node based energy cost metric will capture the real energy consumption in MANETs rather than a link based energy metric. In SS-SPST-F, the dynamic nature of the cost will sometimes make the protocol never stabilize. To show this effect consider the following example given in figure 3. Node R is the parent and it has nodes A and B as its children. Let us assume node B is the farthest neighbor and takes the cost of transmission and reception energy and node A just has the reception energy as its cost and further nodes A and B are neighbors. After all the three nodes establish their respective costs, from the figure we see that node B would be better off if it has

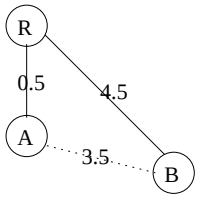

Figure 3. Cost Metric - Unstability example

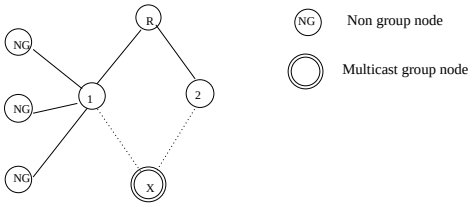

Figure 4. Cost Metric - discard energy wastage

node A as its parent rather than node R since cost to reach R through node A is less than its current cost. If node B makes a move to change its parent to node A, then node A will become the farthest neighbor and the cost from node R to reach node A will increase. It may so happen that, this cost is more than the cost of node B directly connected to node R, so node B will again change its parent and will become the farthest node. This switching of parents will make the protocol unstabilize. So we have to come up with a node metric which stabilizes.

None of the above metrics consider energy expended by non-group nodes in discarding the packet. When there are very few group nodes in the network as opposed to many non-group nodes, which we believe will mostly be the case in a real scenario, the energy spent by the non-group nodes to discard a packet will play a significant role in the total energy consumption of a network. Hence, we need a metric which takes the discarding energy into consideration.

Consider the example given in figure 4. Assume node X wants to join the multicast group and it has to make decision between node 1 and node 2 as its parent node in the multicast tree. Further, assume that the costs to reach the root node R through both the nodes 1 and node 2 are the same. In this example, total energy consumed will be reduced if node X chooses node 2 as

its parent since node 1 has three non-group nodes as its neighbor and every time node 1 transmits, those three nodes will have to spend energy to receive the packet and discard it. Our cost metric tries to minimize this discard energy.

In our system model we assume that when a node transmits all of its neighbors receive and decide to use or discard the information based on their group membership. Thus, when a node transmits a data packet to its farthest group neighbor, all the non-group nodes in the transmission range will receive and discard it. Our cost metric captures this transmission energy and the energy expended by all non-group nodes in discarding the packet. This metric is a variation of SS-SPST-F in that it accounts for the discard energy too and this is a node based metric rather than a link based metric. We believe that node based metric will characterize the energy expended in a better way than a link based metric because energy is associated with each node rather than with each link. Let T_{ij} be the transmission energy required from node i to node j and let R be the reception energy and L_i be the discard energy spent by nodes to discard a packet when node i transmits to its farthest neighbor node j. Let $Neighbors_i$ set represent the neighbor list of node i and $TreeNeighbors_i$ represent the list of neighbors who are supposed to receive the packet from node i. Now the total discard energy for node i, L_i can be presented as follows

$$L_i = R \times (Neighbors_i - TreeNeighbors_i)$$

We can now represent our cost metric C_i as follows

$C_i = T_{ij} + L_i + R$ if j is the farthest node

$C_i = R$ otherwise

We name the protocol with the proposed cost metric, given above, as SS-SPST-E. From the above equation we can see that the source node is assigned a cost based on its current transmission range and the wastage energy of the nodes in this transmission range. Further, we plan to make local decisions based on the least energy cost rather than making a additive cost decision as in SS-SPST.

One possible drawback in our metric is that it does not take remaining energy level into consideration. We have modeled our network nodes to operate in two states, viz. active participant and passive participant. A node can choose to be an active participant, i.e. advertise about its existence and is ready to forward/receive data packets or can choose to be a passive participant, i.e. it just passively receives packets destined for itself. Thus, when a node's remaining energy drops below a certain threshold, it switches to a passive mode and stops sending beacon messages. In the passive mode, a node's existence is not known to its neighboring nodes and hence, it acts just as a receiving node. In this way we take care to prolong the lifetime of the network.

1. SS-SPST-E Algorithm

Each node $i \in V$ maintains a local variable C_i to denote the current energy cost which is calculated as explained above. In addition, each node i maintains a hop count variable $H_i(r)$ to denote the hop count to the root node and a parent variable P_i to denote the current parent to the root. Let us denote the set of all nodes adjacent to node i as $Adj(i)$. A valid value to the variable P_i points to a node which is in the set $Adj(i)$. Each node calculates the overhead cost involved to join a particular node in the tree and tries to join the node with the least overhead cost. Let O_i denote the current overhead cost estimated at node i and $O_i(k), k \in Adj(i)$ is the overhead cost of node i joining node k.

Overhead cost of node i to join node k is calculated as $O_i(k) = \Delta C_k^i$, where ΔC_k^i represents the cost difference experienced by node k with and without node i as its child. The total cost of a tree is given by the sum of cost of all nodes in the tree. We introduce a constant $FLOOR$, which gives the minimum possible value for the total cost of the tree. When all the nodes are in the most energy-efficient path the total cost of tree should be equal to the value of $FLOOR$. Each node in the network, when it is not connected to the tree has a cost as $EMax$, which is greater than the maximum possible cost of the tree. To control the maximum number of hops a node can be from the root we fix the maximum number of hops to be the total number of nodes $NCOUNT$. Let O_i^{min} represent the minimum possible value of O_i.

Remark 3. *For all i, O_i^{min} is determined by the topology of the graphs and group information.*

Let the set $\mathcal{VP}(i)$, for any node i in the graph be defined as,

$$\mathcal{VP}(i) = \{j \mid j \in Adj(i) \wedge H_j(r) < NCOUNT\}.$$

The set $\mathcal{VP}(i)$ contains neighboring nodes of i that have a hop count less than the maximum value.

Let the set $\mathcal{N}(i)$, for any node i in the graph be defined as,

$$\mathcal{N}(i) = \{j \mid j \in \mathcal{VP}(i) \wedge O_i(j) = \min_{k \in \mathcal{VP}(i)} (O_i(k))\}.$$

The set $\mathcal{N}(i)$ contains neighboring nodes of i that are on currently estimated energy-efficient paths from node i to r. Now the legitimate state of the system can be defined as follows

Definition 3. *The system is in the legitimate state iff*

$$\forall i \in V : O_i = O_i^{min} \wedge H_i \leq NCOUNT \wedge ((i = r \wedge P_i$$

$= NULL) \vee (i \neq r \wedge P_i \in \mathcal{N}(i)))$

Any state which is not a legitimate state is an illegitimate state. SS-SPST-E algorithm tries to bring an illegitimate state to a legitimate state and there by reducing the total energy consumption. Each node looks at the states of its negihbors and checks the local satisfiability and if the node is not locally legitimate, it triggers self-stabilization routine and tries to bring the whole system to a legitimate state. This is very similar to SS-SPST. Formally we can describe SS-SPST-E as follows

$$
\begin{aligned}
&\text{if } (i = root \wedge (H_i \neq 0 \vee P_i \neq NULL)) \\
&\text{then } H_i = 0 \& P_i = NULL \& O_i = 0 \\
&\text{else if } \Bigg(i \neq r \wedge (O_i \neq \\
&\qquad \min_{\forall j \in \mathcal{VP}(i)} (O_i(j)) \vee P_i \notin \mathcal{N}(i)) \Bigg) \\
&\text{then } O_i = \min_{\forall j \in \mathcal{VP}(i)} (O_i(j)) \& P_i = j, j \in \mathcal{N}(i) \& H_i = H_j + 1
\end{aligned}
$$

2. Correctness

To prove the correctness of the algorithm we have to prove the convergence property, closure property and the correctness of the algorithm to maintain a single connected tree at any time.

Lemma 5. *Convergence: Starting from a given illegitimate state the total cost of the graph reduces after every round till all the nodes in the system are stabilized.*

Proof. We prove the above lemma using induction. When the node is not connected to the tree it has a cost of $EMax$. After the first round, root node stabilizes and sets its cost variable. Since

$EMax > C_r$, the total graph cost reduces after first round. Now let us assume the cost of the graph after k rounds is TC_k. We now have to prove that if TC_{k+1} is the cost of the graph after $k+1$ rounds then $TC_{k+1} < TC_k$. There are two possibilities,

Case:1 There exists at least one node that is not connected to the tree.

By our assumption, the graph is connected. So there will be at least one tree neighbor to the not connected nodes and in the $k+1^{st}$ round, it will be connected to the tree and its cost will reduce from $EMax$ to a lesser value. Hence, $TC_{k+1} < TC_k$.

Case:2 All nodes are connected to the tree. After k rounds, each node will have a parent in the tree and will have a cost associated with itself. In the $k+1^{st}$ round, when a node changes its parent then the overhead associated with that node reduces. The reduction of cost of the old parent will be greater than the increase of cost of the new parent. Hence, $TC_{k+1} < TC_k$.

Thus the cost of the graph will keep reducing till it reaches a minimum value, which in our case is $FLOOR$. □

Lemma 6. *Closure: Once the cost of the graph equals $FLOOR$, it stays in that value until further fault occurs.*

Proof. If the total cost of the graph decreases after it attains the $FLOOR$ value, then our assumption that $FLOOR$ is the minimum possible energy tree value does not hold true. Hence the total cost of the graph cannot reduce further than $FLOOR$ value. The total cost of the graph will never increase after a round as shown by Lemma 5. Hence, the total cost of the graph will neither decrease nor increase after it attains the $FLOOR$ value. □

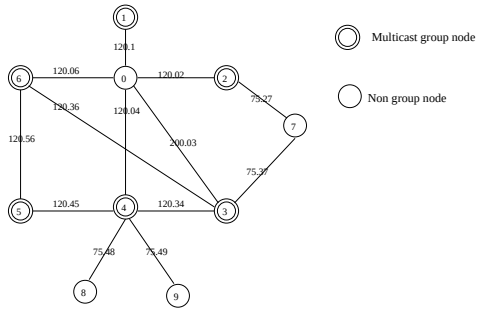

Figure 5. Original Network

Lemma 7. *Count-to-infinity: Tree will always remain connected and there will no loops.*

Proof. There is a possibility that a loop may be created during the tree formation. As soon as a loop forms the hop count value for each node to the root will increase with every round and soon it will exceed the $NCOUNT$ value and hence the loop will be detected and broken. Since by our assumption all the nodes are connected in the graph, the resultant tree will also remain connected as explained in Lemma 5. □

3. Example

We explain the following example to show the working of different energy metrics used in SS-SPST, SS-SPST-F, SS-SPST-T and SS-SPST-E. For theoretical purposes to compare different protocols, we calculate the self-stabilization time in rounds and the total energy consumed to transmit a bit from the source node to all the multicast group nodes.

Figure 5 shows the initial connections of the mobile network. Neighboring nodes are connected using straight lines. Node 0 is the multicast source node and nodes with double circles are multicast-group nodes. The edge weights denote the distance between the two nodes connecting it.

Figure 6 shows the stabilized multicast tree using hop count metric.Here we see, SS-SPST protocol takes 3 rounds to stabilize with nodes in each level stabilizing at each round starting from

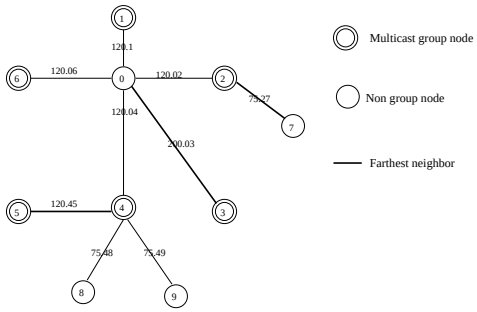

Figure 6. SS-SPST - Hop metric

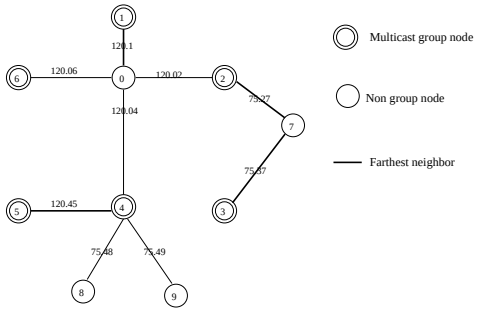

Figure 7. SS-SPST-T - Transmission energy metric

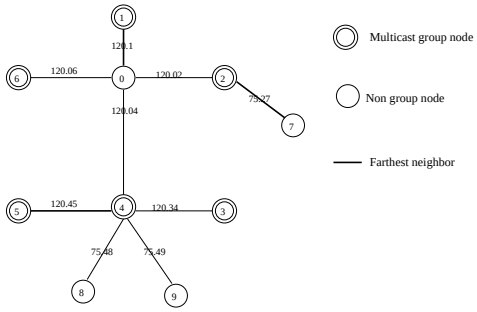

Figure 8. SS-SPST-F - Farthest group node transmission energy

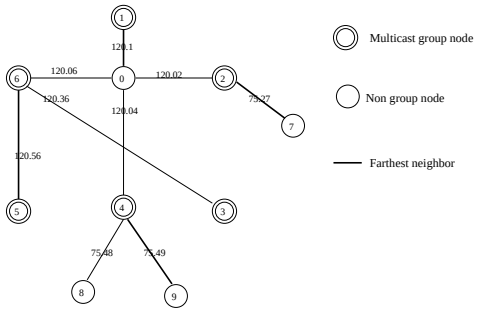

Figure 9. SS-SPST-E - Farthest group node transmission energy with discarding energy

the source node. The total energy consumed by all the nodes in the network to transmit a single bit from the source node to all the multicast destination is the sum of all the transmission energy, reception energy and discard energy. Node 0 has to transmit with certain power to reach its farthest group neighbor node 3. Similarly, node 4 transmits data to its farthest group neighbor node 5. The total energy consumed is equal to $5.9512\mu J$.

Figure 7 shows the multicast tree after stabilization using SS-SPST-T protocol. After round 3, node 3 discovers it has a neighbor node 7 through which it has a more energy-efficient path than its current one. So node 3 leaves node 0 and changes node 7 as its parent. Thus, this protocol takes one more round than SS-SPST to stabilize. From the energy perspective, There are four transmissions involved to reach all the multicast group nodes. Calculating the total energy consumed to transmit a bit based on our energy model in 3 we get, $4.7279\mu J$.

Figure 8 shows the multicast tree built using SS-SPST-F protocol. This protocol forms the multicast tree based on the farthest neighbor and is a realistic depiction of energy consumption. The farthest node bears the cost of transmission while other nodes in the range have just reception energy as their cost. After the first 3 rounds, node 3 will leave node 0 as its current parent and will join node 4 since the cost of joining node 4 will be very low as node 5 is the farthest neighbor of node 4. After node 3 which is a former farthest neighbor makes a move to change its parent in round 4, node 1 in the transmission range of node 0 becomes the farthest neighbor and costs are adjusted based on the new farthest neighbor in round 5. Thus this protocol takes 5 rounds to stabilize. The total energy consumed by the tree to transmit a bit of data to all multicast group neighbors will be equal to $3.3922\mu J$ since there will be only two transmissions.

Finally, figure 9 shows the multicast tree with SS-SPST-E, our proposed, protocol. It tries to build a tree similar to SS-SPST-F based on farthest neighbor at the same time tries to minimize the discard energy expended by non-group neighbors. In figure 8, whenever node 4 transmits a data

packet, nodes 8 and 9 listen to it and discard it. It will be a better idea for nodes 5 and 3 to join node 6 instead of node 4. This takes the same stabilization time as SS-SPST-F and the total energy consumed will be equal to $3.2959\mu J$. Thus we see there is a small but significant energy saving when we choose a path which has less number of non-group nodes.

The following table summarizes the stabilization rounds and the energy spent to transmit a single bit from the multicast source to all the multicast receivers with all the discussed protocols.

Protocol	Stabilization Time in rounds	Energy spent in μ J
SS-SPST	3	5.9512
SS-SPST-T	4	4.7279
SS-SPST-F	5	3.3922
SS-SPST-E	5	3.2959

Table 1. Summary of different metrics for SS-SPST

Thus from the table 3 we see that SS-SPST-E saves more energy than the other protocols but takes more stabilization time than the others. In the following chapters we will simulate the working of all energy metrics and analyze the simulation results.

CHAPTER 6

Simulation Model

To analyze the performance of SS-SPST protocol and analyze the impact of proposed cost metric on it, we simulate SS-SPST, SS-SPST-E, MAODV and ODMRP protocols in a diverse scenario. This section explains the model we used to simulate these protocols and the performance metrics we measured. We used the network simulator (ns-2.1b9a) to compare the performance of the protocols. Our main of this simulation is to bring out the cases where self-stabilizing protocols can be better and worse than other multicast routing protocols. Further, we want to analyze various trade-offs involved in saving energy by simulation results.

We modeled a 750m x 750m simulation area with 50 nodes placed at random positions. Random wave-point mobility model was used in our simulations. We used various scenario files with different initial node locations and different mobility rates and took an average value to plot the graphs. Each simulation ran for 1800 seconds of simulated time. For fairness, we used the same scenarios to evaluate all the protocols.

We assumed each node is equipped with a omni-directional antenna which can dynamically vary the transmission power. From the given nodes, we chose one node to be the source of the multicast session sending CBR data packets at the rate of 64Kbps (16 packets of size 512 bytes, in a sec).

0.1. Metrics. We measure the performance of different protocols based on the following metrics.

1. *Packet delivery ratio*: The ratio of the number of data packets actually delivered to the receivers and the number of data packets supposed to be received by the receivers. This metric shows the effectiveness of a protocol to deliver data packets to multiple recipients. Closer the packet delivery ratio value to 1, better the performance.

2. *Energy consumed per packet delivered*: We measure the energy consumed per packet delivered by calculating the ratio of total energy consumed by all the nodes in the network and the total number of data packets delivered.

3. *Delay per packet delivered*: Delay per packet delivered for a node is calculated by average delay the node experiences in receiving data packets from the sender. An average value for all the nodes in the network describes the delay experienced by the nodes in the network.

4. *Multicasting Overhead*: Multicasting overhead is determined by measuring the control bytes needed to deliver one byte of data successfully. This metric depicts the cost of multicasting. We calculate the multicasting message overhead by calculating the number of control bytes transmitted per data byte delivered. Control bytes not only depict the size of control packets but also the size of data packet headers.

5. *Unavailability Ratio*: Unavailability ratio for a node is the ratio of the multicast service unavailable time of that particular node and the total simulation time. Unavailability ratio depicts the service unavailability time for a particular node.

Our main aim of these simulations is to analyze the performance of protocols based on the above-mentioned parameters by varying mobility speed and group size. Moreover, to analyze the

impact of beacon interval on the protocols SS-SPST and SS-SPST-E we varied the beacon interval and measured the performance metrics. For the first set of simulations we fixed the beacon interval to be 2 seconds and varied the maximum mobility speed from 1-20 m/s and for the second set of simulations we fixed the maximum mobility speed to be 5 m/s and varied the beacon interval from 1 second to 4 seconds.

CHAPTER 7

Simulation Results

Based on the simulations, we analyze the results based on the performance metrics given in the chapter 6.

1. Comparison of different cost metrics

In order to evaluate the performance of SS-SPST with different cost metrics, we simulated the protocols SS-SPST, SS-SPST-T, SS-SPST-F and SS-SPST-E and analyzed the results under diverse scenarios. Our main aim of comparing different energy metrics is to bring out the trade-off involved in saving energy using SS-SPST. We first discuss the results and conclude this section listing out the cost involved in saving energy.

1. **Packet Delivery Ratio** Packet delivery ratio captures the throughput experienced at the user side. Mobility causes nodes to change their locations and hence change in the network topology. Our main aim here is to capture how well the protocols adapt to the rapid changes in topology caused due to mobility of underlying nodes.

   From the figure 10 we see that as the mobility speed of individual nodes increases the packet delivery ratio drops for all the protocols. In the context of dynamic systems, self-stabilization

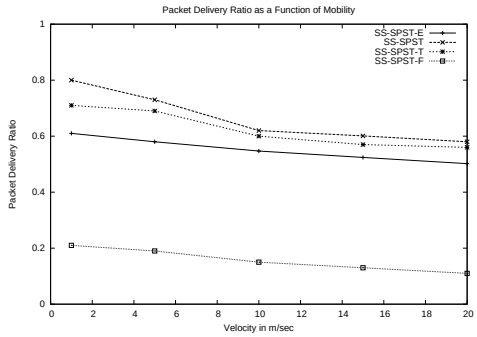

Figure 10. Packet Delivery Ratio as a Function of Velocity

refers to the stabilization time after the ”final” failure. Stabilization is only guaranteed *eventually*, there is an inherent assumption in self-stabilizing system that faults eventually stop to occur for the system to stabilize. As the mobility speed increases, the faults occur at a faster rate than the stabilization time and hence the packet delivery ratio for all the protocols decreases as the mobility speed increases.

Our energy metric forces the SS-SPST-E protocol to choose an energy- efficient path available. Often, transmitting a packet in a single hop might consume more energy than relaying it along a tandem of nodes in a straight line. Thus, the SS-SPST-E tree is deeper than the SS-SPST tree. This reason contributes to reduction in PDR value for SS-SPST-E protocol when compared to that of SS-SPST. The reason SS-SPST-F has a very low PDR value is because of its dynamic nature which causes unstability. From the graph we see that pdr value of SS-SPST-T is slightly better than SS-SPST-E because changes to the tree structure cause changes to the existing cost and hence the protocol needs more time to stabilize.

2. **Energy Consumption** Our main aim of this work is to reduce the energy consumed by SS-SPST protocol by introducing a new cost metric. We measure the amount of energy that can be saved when nodes are operating at different mobility speeds.

   Energy consumption per packet of the nodes is mainly due to transmission and reception of

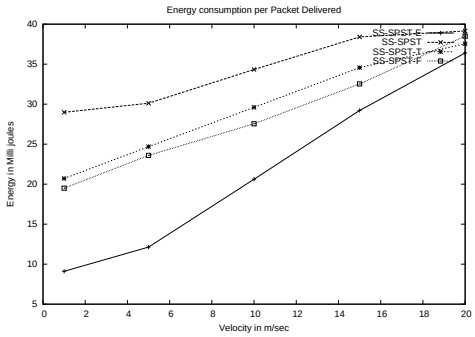

Figure 11. Energy Consumption as a Function of Velocity

control and data packets. The energy spent to transmit control packets is the same at different mobility speeds. As we see from the figure 11, the energy consumed per node in the network drops as the velocity increases. As the velocity increases, the number of faults that occur in the system increases. Thus the total number of packets delivered decreases. Hence we see a decrease in total energy consumed for all the protocols as the velocity increases.

Further from the graph, we see that SS-SPST has the highest energy consumption and SS-SPST-E has the lowest. The other two protocols SS-SPST-F and SS-SPST-T have lesser energy consumption than SS-SPST with the hop count metric but they do not give the most energy-efficient solution to build the multicast tree. Thus our simulation results coincide with our initial analysis.

When the mobility speed is relatively slow, the rate at which the fault occurs because of mobility is also slow and hence the SS-SPST-E protocol transmits most of its packets through energy-efficient paths. As the mobility speed increases the fault rate increases and hence energy saving of SS-SPST-E compared to that of SS-SPST decreases. At high mobility speeds both the protocols consume almost the similar amount of energy per data packet delivered.

3. **Unavailability Ratio** Figure 12 shows the changes in the multicast unavailability ratio with respect to the velocity of nodes. Multicast unavailability ratio captures the time for which a

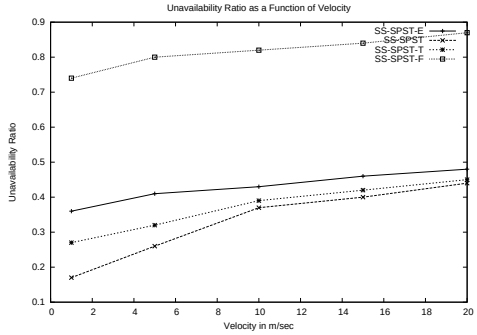

Figure 12. Unavailability Ratio as a Function of Velocity

node remains unconnected to the tree because of link breakages. As the velocity increases, the number of links that break increases in turn increasing the unavailability ratio.

From the figure 12 we see that the unavailability ratio increases as the speed with which the nodes move increases. Self-stabilization protocols rely on the best-effort service. During stabilization phase, a node might not receive packets correctly and unavailability ratio is proportional to the stabilization time. Hence, as the velocity of nodes increase, the stabilization time increases and hence the unavailability ratio. The unavailability ratio of SS-SPST-E is the highest because it takes more time to stabilize as explained previously.

From the above discussion we see that SS-SPST-E is the most energy-efficient among the other protocols. Though incorporation of the new energy metric causes a slight drop in the packet delivery ratio and slight increase in unavailability time, the amount of energy saved per packet is still higher than SS-SPST and other metrics. Our conclusion is that SS-SPST-E is a better choice in a energy constrained networks like MANET than SS-SPST.

2. Beacon Interval

One important parameter in evaluating SS-SPST and SS-SPST-E is the beacon interval. In this section we analyze the impact, beacon interval has a performance of the protocol.

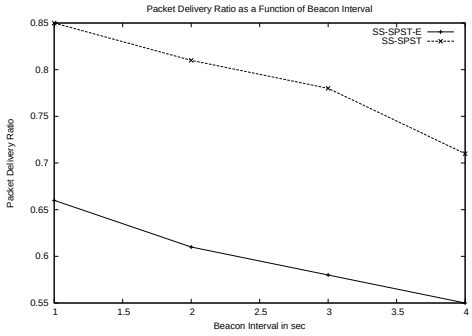

Figure 13. Packet Delivery Ratio as a Function of Beacon Interval

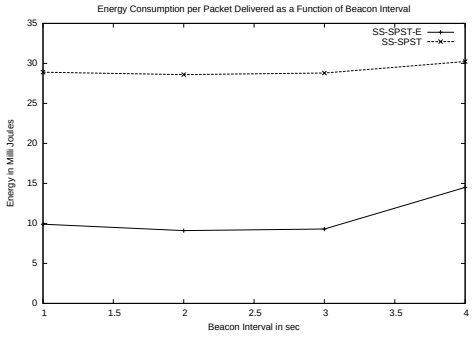

Figure 14. Energy Consumption as a Function of Beacon Interval

1. **Packet Delivery Ratio** Figure 13 shows the variation of packet delivery ratio (PDR) with beacon interval. As the beacon interval increases the PDR value drops for both the protocols, SS-SPST and SS-SPST-E. This drop is seen due to the fact that time to realize that a fault has occurred in the system increases with the beacon interval. With low beacon interval values the faults are identified and corrected quicker and hence we see a higher packet delivery ratio. From the graph, we see that the decrease in PDR value is linear till a point and then drops at a faster rate. This fact is because the increase in beacon interval makes some nodes to move out of range completely. In other words, number of faults that occurs when beacon interval is more than 3 seconds increases. Hence, we see a sharp drop after beacon interval of 3 seconds. From this we conclude, for better packet delivery ratio results we can fix a low value for the beacon interval and a value above 3 seconds will not depict the real performance of SS-SPST.

2. **Energy Consumption** Figure 14 shows the variation of energy consumption with beacon interval. Energy consumption per packet delivered value is altered by total energy consumption and number of packets delivered. As the beacon interval increases, one expects the energy consumed to decrease as the number of control packets sent will decrease. From the graph we see that the energy consumed per packet delivered value decreases for a while and then increases. This increase is attributed to the reduction in the number of packets delivered with increase in beacon interval as explained by figure 13. At higher velocities the rate of decrease in packet delivery ratio is more than the rate of increase in energy saving thus energy expended per packet increases beyond certain velocity.

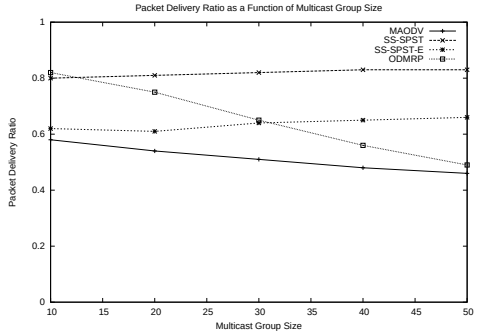

Figure 15. Packet Delivery Ratio as a Function of Multicast Group Size

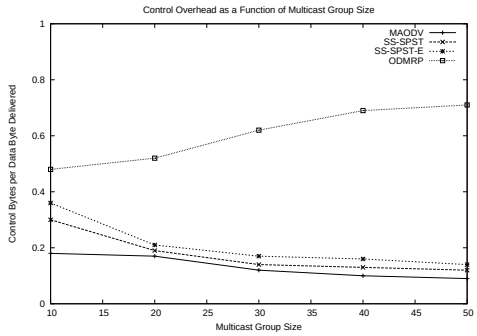

Figure 16. Control Byte Overhead as a Function of Multicast Group Size

Use of short beacon intervals increases number of control messages and use of long beacon intervals decreases the packet delivery ratio. Our conclusion is that for beacon interval value around 2 seconds, we have an optimum energy consumed per packet delivered. Hence, in all our simulations we will be using a beacon interval of 2 seconds.

3. Comparison with other MANET protocols

In this section we compare SS-SPST and SS-SPST-E protocols with other MANET protocols like MAODV and ODMRP. We analyze the comparison results and bring out the scenarios when self-stabilizing algorithms will be useful over on-demand multicast protocols.

1. **Packet Delivery Ratio**

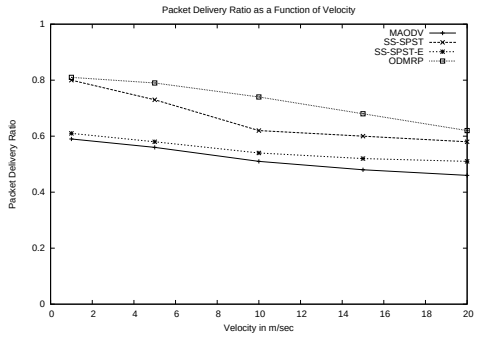

Figure 17. Packet Delivery Ratio as a Function of Velocity

We varied the multicast group size to calculate the packet delivery ratio. Each node in the network was made to move with a minimum speed of 0m/s and with a maximum speed of 1m/s. The multicast group size was varied from 10 to 50 members in steps of 10.

From the graph (figure 15) we see that self-stabilizing protocols are group-scalable i.e., when the multicast group nodes are more, the self-stabilizing protocols perform as well as they do when the multicast group nodes are less. In other words self-stabilizing protocols scale well with increase in the number of multicast group nodes.We see that packet delivery ratio is almost the same for different set of multicast nodes.

ODMRP outperform self-stabilizing protocols when the number of multicast nodes is less. It does not scale well with the increase in the multicast group nodes. As we see from the graph the packet delivery ratio dropped drastically as the number of multicast nodes increases. This decrease in packet delivery ratio is attributed to the increase in overhead caused due to the building of redundant paths.

Of all the four protocols, packet delivery ratio of MAODV is the least. Though packet delivery ratio is not affected too much by the increase in number of multicast nodes, it is less when compared to other protocols

We varied the multicast group size to calculate the multicast control message overhead. Each

node moves in the network with a minimum speed of 0m/s and a maximum speed of 1m/s. The multicast group size was varied from 10 to 50 members in steps of 10.

The graph (figure 16) depicts the cost involved in multicasting using different protocols. This figure represents the control byte overhead when the data is sent continuously.

Since ODMRP is a mesh-based protocol, it uses more control bytes for building the multicast tree. ODMRP constructs more than one route from source to destination and hence uses many control messages. As the number of group members increase, ODMRP behaves similar to *flooding* algorithm and hence the control byte overhead increases with increase in number of group members.

Of all the four protocols, MAODV uses the least control byte overhead It sends multicast messages only when there is a need for multicasting.

SPST and SS-SPST-E are proactive multicast protocols. The number of control packets sent by these two protocols is almost the same. When multicasting of data packet takes place continuously, these protocols have a very less control bytes overhead but when there is no multicasting, these protocols still send out control packets to maintain the tree structure. Thus these self-stabilizing protocols will have higher multicasting overhead when there is no multicasting operation. Further, SS-SPST-E sends additional information in its beacon packet to conserve energy and hence has more control byte overhead than SS-SPST.

We varied the mobility speed of the nodes in the network from 1m/s to 20 m/s while fixing the multicast group size at 20. By varying the mobility we measured the packet delivery ratio.

Mobility against packet delivery ratio graph (figure 17) shows the performance of the multicasting protocols in presence of mobility. We see that packet delivery ratio in all the protocols drops as the mobility of the nodes increases.

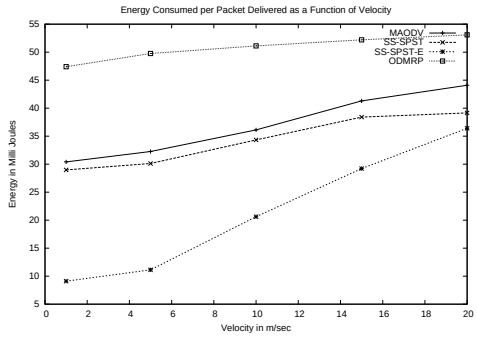

Figure 18. Energy Consumption per Packet Delivered as a Function of Velocity

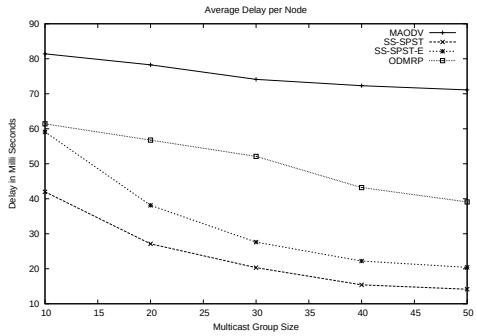

Figure 19. Average Delay as a Function of Multicast Group Size

Self-stabilizing protocols take finite time to build a multicast tree. As the underlying nodes move, the multicasting tree should be reconstructed. From our simulations, we found that SS-SPST-E takes more time to build a tree than SS-SPST. This is due to the fact explained previously in section 1.

ODMRP's packet delivery ratio is the highest among the protocols compared even at high mobility speeds. By providing alternate paths, ODMRP performs better than other protocols even under high mobility. MAODV, which is a tree-based protocol performs in a similar fashion to SS-SPST. As the mobility speed increases the packet delivery ratio decreases.

2. **Energy Consumption per packet delivered**

By varying the mobility speed we captured the energy spent to successfully deliver a packet

to the receiver in all the protocols. Our aim was to capture the fact how energy-efficient is SS-SPST-E when compared to other protocols.

From the figure 18 we see that ODMRP has the highest energy consumption per packet and SS-SPST-E has the least. Energy is expended on transmission of control messages to build multicast structure and data messages along the route. ODMRP constructs more than one path to deliver the data message to the receiver and hence consumes maximum energy among all the other protocols. MAODV and SS-SPST construct tree structure to deliver multicast messages. Although SS-SPST has more control overhead than MAODV, the packet delivery ratio of SS-SPST is also more. Hence, SS-SPST spends little less energy per packet delivered than MAODV. An interesting point to note is that SS-SPST-E expends the least energy than any other protocol. When the mobility speed is not too high, we have maximum energy conservation using SS-SPST-E.

3. **Delay** We measure the average delay experienced per packet delivered from source for all the four protocols to find out the protocol which is suitable in a delay constrained network. Proactive protocols build the multicast structure even before the multicast data messages arrive as opposed to on-demand protocols which build the multicast structure based on demand. We measured the average delay by varying the number of group members.

   From the figure 19 we see that the delay for ODMRP and MAODV, on-demand multicast protocols, is more than SS-SPST and SS-SPST-E, proactive multicast protocols. By maintaining multiple routes to multicast receivers, ODMRP has a delay value lesser than MAODV. When number of group members are high, ODMRP behaves like *flooding* protocol and hence rate of decrease of delay value in ODMRP is greater than MAODV.

   SS-SPST-E and SS-SPST have delay values lesser than their on-demand counterparts. How-

ever, SS-SPST-E has a slightly more delay value than SS-SPST due to the increase in stabilization time. Thus, increase in delay is another trade-off apart from reduction in packet delivery ratio by adopting the proposed energy metric for SS-SPST.

CHAPTER 8

Implementation

Many multicast algorithms for MANETs have been proposed earlier in the literature. However, most of these studies have been simulation based, with few real implementation. Validating MANET multicast protocols with real implementation is necessary for their usage in real world. Though simulation results to some extent characterize the correctness of the protocol, they do not provide the full picture. We believe, only a real implementation of protocols will give a more accurate performance results. Traditionally, implementing MANET protocols have been difficult because of the cost and time involved in developing and testing. With the present development in hardware technology, the cost and time in development have considerably reduced.

Java Self-Stabilization (JASS) is an application-level implementation of SS-SPST protocol. It can be used with IEEE 802.11 cards configured in ad hoc (Independent Basic Service Set - IBSS) mode. We chose Java as the implementation language to achieve portability across different platforms. We wanted to develop an implementation which does not require too many modifications to the kernel and still capture the characteristics of the real world. So we chose to develop SS-SPST at application level and bypass the routing and multicasting mechanisms of the underlying network.

In our implementation, we broadcast all the packets and they have a time-to-live (TTL) value as 1. Thus, the packets travel only single hop and the JASS protocol at the application layer decides whether the packet has to be forwarded or not. By doing so we bypass the underlying routing and

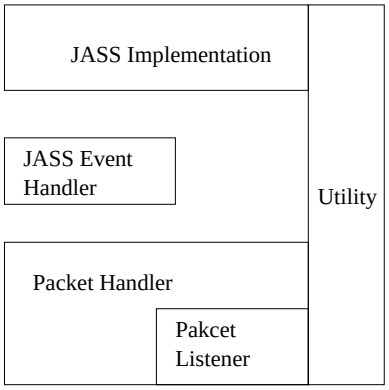

Figure 20. Java Self-Stabilization Implementation Architecture

multicasting protocols used. We maintain a configuration file to configure dynamic properties like multicast port, beacon interval etcetera.

JASS architecture is based on the architecture used in [19] to develop an implementation of ODMRP. JASS has five main components as given in figure 20

1. **Packet Listener:** This module is responsible for receiving packets from the network. It is run as a separate thread listening to a pre-configured multicast port. Whenever this module receives a packet, it passes on to the above layers.

2. **Packet Handler:** Packet Handler is responsible for all packet related sending and receiving. SS-SPST protocol sends beacon and data packets through this module. It also has the logic to forward or drop the packets received from Packet Listener module.

3. **Event Handler:** Event Handler module is an asynchronous event module which processes events from Packet Handler. Whenever Packet Handler decides to pass a packet to the upper layer, an event is generated and this module decides further actions of the packet. If the received packet is a valid packet then event handler sends the packet to the protocol layer where the packet will be processed.

4. **Protocol Layer:** Protocol Layer forms the logic of the SS-SPST protocol. It registers itself with the Event Handler to be notified of any new valid packet arrival and uses Packet Handler to send broadcast packets.

5. **Utilities:** This module consists of many utility classes like timers, log, configuration loader etcetera. These classes are mostly Java static classes and can be used by all the other modules.

1. Network Configuration and Experiments

As an initial attempt to experiment with the real implementation of SS-SPST, we used the 3 java-enabled laptops to run JASS on it. The test interfaces were equipped with a 11 Mbps Lucent WaveLAN IEEE 802.11b PC Card (”Silver”). For reference the implementation code is available at [22].

Our main aim of these experiments were to measure the how quickly a node discovers its neighbors and joins a tree in a real scenario. Further we wanted to measure the time a node takes to recover from a fault. Initially, we placed all the hosts in wireless range, hosts were placed in close proximity, i.e. in the same office. We started the JASS protocol in all the hosts and measured the stabilization time. Next stage was to induce a fault caused by movement. To achieve this, we moved a host out of range of one and within range of another. Suitable locations were determined by walking around with laptops running “ping” tests[1]. Hosts intended to have connectivity were placed several meters inside the range at which connectivity began to falter. One particular host was chosen to be the root node. Initially, the other two hosts will have root as their parent since they will be in the range of root node. As we moved one host out of range of root node, we induced a fault and calculated the time it took to choose a new parent. Our results show that the stabilization time is similar to that of the simulated version.

[1]The author would like to thank his colleagues for their support

1.1. Emulation. Measuring stabilization time with 3 laptops does not give the complete real picture. In order to evaluate our implementation we emulated the functionality of wireless networks and tested our protocols. We developed JASS-E, a JASS implementation with energy metric to compare the results. For evaluation purposes, we ran many JASS, JASS-E implementations on various wire-line hosts and emulated mobility. Each node was allowed to move in a specified rectangular area with a pre-defined maximum velocity. We emulated the energy model we have adopted in 3 and induced a bit error rate of 1×10^{-5}. To emularte mobile ad hoc networks, we made the nodes receive packets from only from nodes within a predefined maximum transmission range. All the other packets are discarded at *Packet Listener* layer itself. The following tables show that our implementation results matched with our simulation results.

Velocity m/sec	PDR		Energy in mJ	
	Simulation	Emulation	Simulation	Emulation
0	0.81	0.83	28.98	28.9
5	0.73	0.74	30.12	30.04
10	0.62	0.64	34.33	34.17
15	0.601	0.63	38.41	38.37
20	0.58	0.59	39.15	39.06

Table 2. Comparison of simulation and emulation results of JASS protocol

Velocity m/sec	PDR		Energy in mJ	
	Simulation	Emulation	Simulation	Emulation
0	0.61	0.62	9.11	8.9
5	0.58	0.56	12.13	13.24
10	0.547	0.54	20.62	22.17
15	0.524	0.53	29.21	28.61
20	0.502	0.49	36.4	37.01

Table 3. Comparison of simulation and emulation results of JASS-E protocol

Though our real implementation and emulation capture some aspects of SS-SPST protocol,

a real test would be to test the real implementation with many hosts. In future, we plan to implement JASS in other mobile devices and test it in a network with numerous hosts running on different machines and different platforms.

CHAPTER 9

Conclusion

Self-stabilizing protocols for constructing multicast tree provide a different perspective for multicasting in MANETs. In this work, we implemented SS-SPST algorithm in a distributed manner, proposed an energy-efficient cost metric for the SS-SPST algorithm to conserve energy and performed simulation based performance analysis of SS-SPST with other cost metrics and compared the performance of SS-SPST with other multicast protocols like ODMRP and MAODV. We tested these protocols with different mobility patterns and different multicast group sizes.

A general conclusion is that, self-stabilizing protocols are group-scalable and have a better packet delivery ratio even when the number of multicast group nodes is large. They have less control message overhead compared to mesh based protocol ODMRP but these protocols are pro-active and they use control messages even without any multicasting operation. Thus they might not be an ideal choice when multicasting operation is not often performed. The performance self-stabilizing protocols significantly drops under high mobility condition since the rate at which the faults occurs is faster than the rate at which the system stabilizes. Thus they are well suited for slow and average mobility conditions.

Incorporation of our new energy metric is very energy-efficient at low mobility speed. Both SS-SPST and SS-SPST-E have high packet delivery ratio and energy-efficient cost metric minimizes total energy expended. Thus, this genre of self-stabilizing protocols can be used in situations where,

when complete solution cannot be offered a best-effort solution would be preferred. Our simulation results show that to save energy we need to compromise on the throughput. Thus, there is a cost paid to save energy. The goal of energy-efficient techniques should be minimize this cost as much as possible.

We see that the self-stabilizing protocols will be a suitable solution to use in ubiquitous computing scenarios where the mobility is less and packet delivery ratio should be high. We plan to investigate how we can use self-stabilizing multicast protocols for ubiquitous computing.

Still there is much scope for improvement in these protocols. At present these protocols send out the same size of control packets even when there is no multicast operation thus increasing the multicast overhead. No prescribed ways have been mentioned to calculate the link weights. Thus much room exists to increase the performance and decrease the overhead of these protocols.

REFERENCES

[1] S. K. S. Gupta, A. Bouabdallah and P. K. Srimani. “Self-Stabilizing Protocol for Shortest Path Tree for Multi-cast Routing in Mobile Networks (Research Note)”. *In Proc. of Euro-Par 2000*, May 2000.

[2] P. K. Srimani, S. K. S. Gupta. “Self-Stabilizing Multicast Protocols for Ad Hoc Networks”. *Journal of Parallel and Distributed Computing*, vol. 63, no. 1, pp. 87-96, 2003.

[3] A. S. Tanenbaum. ”Computer Networks”. *Prentice Hall*, ISBN 0130661023, 4th edition, 2002.

[4] A. Ephremides. “Energy Concerns in Wireless Networks”. *In Proc. of IEEE Wireless Communications*, vol. 9, no. 4, pp. 46-59, Aug. 2002.

[5] J. E. Weiselthier, G. D. Nguyen and A. Ephremides. “On the Construction of Energy-Efficient Broadcast and Multicast trees in Wireless Networks”. *In Proc. IEEE INFOCOM 2000*, 2000, pp. 585-594.

[6] J. E. Weiselthier, G. D. Nguyen and A. Ephremides. “Algorithms for Energy-Efficient Multicasting in Static Ad-Hoc Wirele ss Networks”. *In Proc. Journal on Mobile Networks (MONET)*, vol. 6, 2001, pp. 2 51-63.

[7] J. E. Wieselthier, G. D. Nguyen and A. Ephremides. ”Resource Management in Energy-Limited, Bandwidth-Limited, Transceiver-Limited Wireless Networks for Session-Based Multicasting”. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 39, Issue 5, June 2002.

[8] J.-H. Chang, L.Tassiulas ”Energy Conserving Routing in Wireless Ad Hoc Networks”.; *In Proc. of IEEE INFOCOM 2000*, 2000, pp. 22-31.

[9] A. Misra and S. Banerjee. ”Minimum Energy Paths for Reliable Communication in Multi-Hop Wireless Networks”. *CS-TR 4315, Technical Report, Department of Computer Science, University of Maryland, College Park*, December 2001.

[10] E. Royer and C. E. Perkins. ”Multicast operation of the ad-hoc on-demand distance vector routing protocol”. *In Proc. Of the 5th ACM/IEEE Annual Conf. On Mobile Computing and Networking*, August 1999.

[11] M. Gerla, S. -J. Lee and C. -C. Chang. ”On-Demand multicast routing protocol (ODMRP) for ad hoc networks”. *In Proc. Of IEEE Wireless Communications and Networking Conference 1999*, New Orleans, LA, September 1999.

[12] S. Ni, Y. Tseng, Y. Chen and J. Sheu. “The Broadcast Storm Problem in a Mobile Ad Hoc Network”. *In Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Seattle, Washington, 1999.

[13] E. W. Dijkstra. “Self Stabilizing systems in spite of distributed control”. *In Proc. Communications of the ACM*, November 1974.

[14] E. W. Dijkstra. “A Belated Proof of Self-Stabilization”. *Distributed Computing*, 1986.

[15] S. Ramanathan. “Multicast Tree Generation in Networks with Asymmetric Links”. *IEEE/ACM Transaction on Networking*, vol. 4, no. 4, pp. 558-568, November 1996.

[16] S. Singh and C. S. Raghavendra. ”PAMAS-Power Aware Multi-Access Protocol with Signaling for Ad Hoc Networks”. *ACM Communications Review*, July 1998.

[17] S. Deering. ”Host Extensions for IP Multicasting”. *RFC 1112*, August 1989, Available from http://www.ietf.org/rfc/rfc1112.txt

[18] Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks (MANET) Working Group Charter. http://www.ietf.org/html.charters/manet-charter.html.

[19] Java On-Demand Multicast Routing Protocol. http://homepages.cs.ncl.ac.uk/einar.vollset/home.formal/jomp.ht

[20] C. Genolini and S. Tixeuil. “A lower bound on dynamic k-stabilization in asynchronous systems”. *SRDS 2002 21st Symposium on Reliable Distributed Systems, IEEE Computer Society Press*, pp. 211-221, 2002.

[21] W. Heinzelman, A. Chandrakasan, H. Balakrishnan. “Energy-Efficient Communication Protocol for Wireless Microsensor Networks”. *In Proc. of 33rd International Conference on System Sciences (HICSS ’00)*, January 2000.

[22] G. Sridharan. JASS Implementation. http://shamir.eas.asu.edu/ mcn.