P-3322

TREE AGGREGATION IN MANET

A PROJECT REPORT

Submitted by

SUGUMARAN S	71206205052
VINOTH KUMAR J	71206205057
VIJAYASHANKAR V	71206205307

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2010

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**Tree Aggregation in MANET**" is the bonafide work of **SUGUMARAN S, VINOTH KUMAR J** and **VIJAYASHANKAR V** who carried out the project work under my supervision.

[illegible] 19/04/10

SIGNATURE

Mrs.J.Cynthia M.E.,

Senior Lecturer,

Department of Information Technology,

Kumaraguru College of Technology,

Chinnavedampatti Post,

Coimbatore - 641006.

[illegible]

SIGNATURE

Dr.L.S.Jayashree,Ph.D.

HEAD OF DEPARTMENT,

Department of Information Technology,

Kumaraguru College of Technology,

Chinnavedampatti Post,

Coimbatore - 641006.

Submitted for viva-voice examination held on 20-4-2010.

[illegible] 20/4/10

INTERNAL EXAMINER

[illegible] 20.4.10

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The exhilaration achieved on successful completion of any task should be shared with the people behind the venture. At the onset, we thank the management of our college for having provided the excellent facilities to carry out the project.

We express our deep gratitude to our Principal **Dr.S.Ramachandran** for ushering us in the path of triumph.

We are always thankful to our beloved Dean and HEAD of Computer Science and Engineering Department, **Dr.S.Thangasamy**, whose consistent support and enthusiastic involvement helped us a great deal.

We convey our sincere thanks to our project coordinator **Dr.L.S.Jayashree**, for her invaluable assistance.

We are greatly indebted to our beloved guide **Mrs.J.Cynthia M.E**, Senior Lecturer, Department of Information Technology for her excellent guidance and timely support during the course of this project.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Information Technology

TABLE OF CONTENTS

ABSTRACT

Tree aggregation is an efficient proposition that can solve the problem of multicast forwarding state scalability and the number of control messages required to maintain them. The main idea of tree aggregation is to force several groups to share the same delivery tree: in this way, the number of multicast forwarding states is reduced. It reduces the number of trees by forcing multiple groups to share the same tree. These groups are said to be aggregated to this tree. By this way, only one forwarding entry per AP is needed for all these groups instead of one per group and per AP. Thus, the number of forwarding entries, depending on the number of trees, is reduced. Moreover, fewer trees are maintained. Different groups may be aggregated to the same tree. Decreasing the number of trees reduces the control overhead due to the maintenance of trees. Tree aggregation consists in building fewer trees are built and less forwarding entries are stored than with traditional multicast while controlling the wastage of bandwidth. In this paper, we propose a study on the number of Adaptive core node election algorithm that need to be configured in a domain depending on the tree construction Algorithm. Our results show that for a given set of multicast groups, even when this set includes all the possible groups, the number of trees that need to be configured is small. These protocols build multicast trees using location information and use geographic forwarding to forward packets down the multicast trees.

LIST OF FIGURES

1. INTRODUCTION

The growth of multi-users applications such as videoconferences,file sharing, chat rooms or multi-playergames is constantly increasing the demand on networkbandwidth.To manage these communications, multicast consists in sending only one message per group instead of unicasting the message to all the members of the group. Each group using different services such as audio-video conferencing, Internet video-games, Internet TV or online teaching.For several years, multicast has been considereda solution to save bandwidth.Tree aggregation has been proposed to reduce both the number of multicast forwarding states and its control overhead. While most multicast protocols build a multicast tree per group, tree aggregation uses significantly less trees than groups. Since the states and control overhead depends mainly on the number of trees, the tree aggregation technique can achieve major savings.

In Tree-based approaches, nodes send their data directly to Tree-head and.Tree-head then aggregate and forward the data to-wards sink. Normally, the tree based structure will take time and energy for their construction and their maintenance. Also, if a non leaf node breaks, the particular sub section of the tree is disconnected from the network. It necessitates the implementation of reliability methods to ensure the delivery of the packets. Due to the dense deployment nature of the sensor networks, it is necessary to take care of the spatial and temporal correlations into account while designing tree aggregation algorithms.The energy consumption is low as compared to that of sending data directly to Tree-head.

Tree aggregation is an efficient proposition that can solve the multicast forwarding state scalability problem. It reduces the number of multicast forwarding states and the tree maintenance overhead. To achieve this reduction, several multicast groups share the same delivery tree within a domain. Consequently, fewer trees are built and less forwarding entries are stored than with traditional multicast. In particular, *efficient* multicast for wireless networks is particularly critical due to the limited energy availability in such networks. These protocols build multicast trees using location information and use geographic forwarding to forward packets down the multicast trees.

2. LITERATURE REVIEW

2.1 Position-Based Multicast [1]

For multicast it is necessary to establish a distribution tree among the nodes, along which packets are forwarded toward the destinations. At the branching points of the tree, copies of the packet are sent along all the branches. Two – potentially conflicting – properties are desirable for such a distribution tree

- the length of the paths to the individual destinations should be minimal and
- the total number of hops needed to forward the packet to all destinations should be as small as possible.

If the topology of the network is known, a distribution tree that optimizes the first criterion can be obtained by combining the shortest paths to the destinations. Wherever these paths diverge, the packet is split. With position-based routing, routing decisions are based solely on local knowledge, thus neither the shortest paths to all destinations can be used directly. Instead PBM uses locally available information to approximate the optima. Given this information the main task of a forwarding node in PBM is to find a set of neighbors that should forward the packet next. We call these neighbors the *next hop nodes*. The current node will assign each destination of the packet to exactly one next hop node. Each next hop node then becomes forwarding node for this packet toward the assigned destinations. If the current node selects more than one next hope node, then the multicast packet is split. This may be required in order to reach destinations which are located in different directions relative to the forwarding node. The most important property of PBM is that each forwarding node autonomously decides how to forward the packet. This decision requires no global distribution structure such as a tree or a mesh. There are two distinct cases that can occur when a forwarding node selects the next hop nodes: either for each destination exists at least one neighbor which is closer to that destination than the forwarding node itself. In this case *greedy multicast forwarding* is used. Otherwise the node employs *perimeter multicast forwarding*.

Position-based routing algorithms eliminate some of the limitations of topology-based routing by using additional information. They require that information about the physical position of the participating nodes be available. Commonly, each node determines its own position through the use of GPS or some other type of positioning service. A *location service* is used by the sender of a packet to determine the position of the destination and to include it in the packet's destination address.

The routing decision at each node is then based on the destination's position contained in the packet and the position of the forwarding node's neighbors. Position-based routing thus does not require the establishment or maintenance of routes. The nodes have neither to store routing tables nor to transmit messages to keep routing tables up to date. As a further advantage, position-based routing supports the delivery of packets to all nodes in a given geographic region in a natural way. This type of service is called *geocasting*.

2.2 Scalable Position-Based Multicast [2]

Many applications envisioned for mobile ad-hoc networks rely on group communication. Messaging during disaster relief, networked games, and emergency warnings in vehicular networks are common examples for these applications. As a consequence, multicast routing in mobile ad-hoc networks has received significant attention over the recent years. The forwarding strategy uses information about the geographic positions of group members to make forwarding decisions. In contrast to existing approaches it neither requires the maintenance of a distribution structure nor resorts to flooding. The group management scheme uses knowledge about geographic positions for a hierarchical aggregation of membership information.

The forwarding of packets by SPBM is a generalization of position-based unicast routing as proposed. In these protocols, a forwarding node selects one of its neighbors as a next hop in a *greedy* fashion, such that the packet makes progress toward the geographic position of the destination. It is possible that a node has no neighbor with progress toward the destination although a valid route to the destination exists. The packet is then said to have reached a local optimum. In this case, a *recovery strategy* is used to escape the local optimum and to find a path toward the destination. The most important characteristic of position-based routing is that forwarding decisions are based only on local knowledge. It is not necessary to create and maintain a global route from the sender to the destination. Therefore, position-based routing is commonlyregarded as highly scalable and very robust against frequenttopological changes. In order to extend position-based routingto multicast, SPBM provides an algorithm for splittingmulticast packets at intermediate nodes when destinationsfor that packet are no longer located in the same direction.This strategy includes both greedy forwarding and the recoverystrategy.

The second important element of SPBM is its groupmanagement scheme. It relies on geographic informationto achieve scalability: instead of maintaining a fixed distribution structure, an intermediate node just needs to knowwhether group members are located in a given direction ornot. This allows a hierarchical aggregation of membershipinformation: the further away a region is from an intermediatenode the higher can be the level of aggregation for thisregion. Therefore, group membership management can beprovided with an overhead that scales logarithmically withthe number of nodes and that is independent of the numberof multicast senders in a multicast group. A second observationis then used to reduce this overhead further: the higherthe level of aggregation the lower the frequency of membershipchanges for the aggregate. In SPBM, we thereforepropose to scale down the frequency of membership updatemessages exponentially with the level of aggregation. Thisresults in a constant upper bound on the overhead as the sizeof the network increases.

2.3 Geographic Multicast Routing (GMR) [3]

Geographic Multicast Routing (GMR), a new multicast routing protocol for wireless sensor networks. GMR manages to preserve the good properties of previous geographic unicast routing schemes while being able to efficiently deliver multicast data messages to multiple destinations. It is a fully-localized algorithm (only needs information provided by neighbors) and it does not require any type of flooding throughout the network. Each node propagating a multicast data message needs to select a subset of its neighbors as relay nodes towards destinations. GMR optimizes cost over progress ratio. The cost is equal to the number of selected neighbors, while progress is the overall reduction of the remaining distances to destinations. That is, the difference between distance from current node to destinations and distance from selected nodes to destinations. Such neighbor selection achieves a good trade-off between the cost of the multicast tree and the effectiveness of the data distribution. Our cost-aware neighbor selection is based on a greedy set merging scheme achieving a O (Dnmin(D, n)3) computation time, where n is the number of neighbors of current node and D is the number of destinations. This is superior to the exponential computational complexity of an existing solution (PBM) which tests all possible subsets of neighbors, and to an alternative solution that we considered, tests all the set partitions of destinations.

Delivery to all destinations is guaranteed by applying face routing when no neighbor provides advance toward certain destinations. Our simulation results show that GMR outperforms previous multicast routing schemes in terms of cost of the trees and computation time over a variety of networking scenarios. In addition, GMR does not depend on the use of any parameter, while the closest competing protocol has one parameter and remains inferior for all values of that parameter.

2.3.1 Greedy Neighbor Selection

Algorithm 1 Greedy set partition selection algorithm

```
1: $M = \{M1, M2, ..., Mm\}$ so that $Mi = \{Dj$ | same neighbor provides most advance$\}$
2: $CRatio = m/(summation\ of\ pi)$.
3: repeat
4: $Best\ Reduction = 0$
5: for all pairs $\{Mi, Mj\}$ do
6: Find cost over progress reduction by merging of $\{Mi, Mj\} \in M$
7: if $reduction > Best\ Reduction$ then
8: $BestReduction = reduction$
9: $Best\ Merge = \{Mi, Mj\}$
10: end if
11: end for
12: if $Best\ Reduction > 0$ then
13: $M = \{M1, M2, ..., \{Mi, Mj\}, ..., Mm\}$
14: $CRatio = |M|$
15: end if
16: until $Best\ Reduction = 0$
```

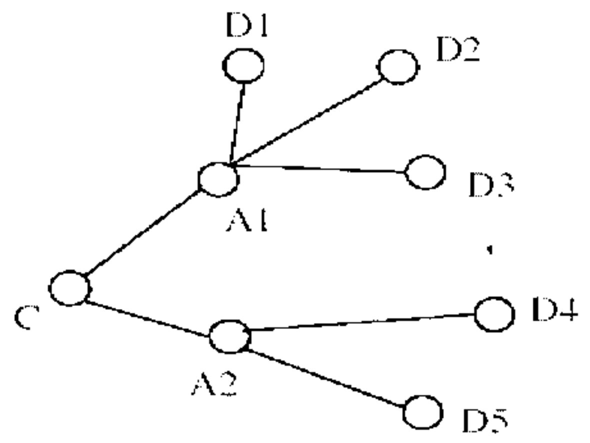


Fig.1 Evaluating the candidate forwarding from C to A1 and A2

2.3.2 Cost over Progress Metric

The cost over progress metric can be used to select the next hops towards destinations. For clarity we first introduce it for the unicast case, and we will later explain how we extend it to multicast scenario. In a unicast scenario, node *C*, currently holding the packet, will forward it to its neighbor *A*, closer to the destination *D* than itself, that minimizes the ratio of cost over progress. The cost function depends on the assumptionsand metrics used, while progress measures the advancetowards the destination.

Considering the multicasting problem, where a sourcenode wishes to send a packet to a number of destinationswith known positions. It is assumed that the numberof such destinations is small, which is a reasonable assumptionfor the considered scenarios. Unlike PBM, we describehere a solution which does not need any parameter. Assumethat a node *C*, after receiving a multicast message, is responsiblefor destinations *D*1,*D*2 . . . *D*5, and that it evaluatesneighbors *A*1,*A*2 as possible candidates for forwarding. Thewhole task can be sent to a single neighbor or can besplit to several neighbors, each with a subset of destinationsto handle. Hop count is assumed to be proportional todistances.

The current total distance for multicasting is

$$T1 = |CD1| + |CD2| + |CD3| + |CD4| + |CD5|$$

IfC considers $A1$ and $A2$ as forwarding nodes, covering$D1, D2, D3$, and $D4, D5$, respectively,the new total distanceis

$$T2 = |A1D1|+|A1D2|+|A1D3|+|A2D4|+|A2D5|$$

andthe progress made is $T1 - T2$. Our aim is also to minimize theconsumption of bandwidth, which is proportional to the totalnumber of forwarding nodes selected. Thus, the cost is thenumber of selected neighbors, which in the above example are2.. Thus the forwarding set $\{A1, A2\}$ is evaluated as $2T1 - T2$.Among all candidate forwarding sets, the one with minimalvalue of this expression is selected.

2.4 Hierarchical Rendezvous Point Multicast (HRPM) [4]

Hierarchical rendezvous point multicast (HRPM) reduces the encoding overhead by employing two key design concepts:

- Use of hierarchical decomposition of multicast groups and
- Leveraging geographic hashing to construct and maintain such a hierarchy efficiently.

The main design goal of HRPM is to limit the per-packet overhead to an application-specified constant (x), irrespective of the group size G. It achieves this by recursively partitioning a large multicast group into manageable-sized subgroups in which the tree-encoding overhead satisfies the x constraint. This partitioning is achieved by geographically dividing the deployment area into smaller and smaller cells, which form a hierarchy with the root representing the entire region. Every cell in the hierarchy has an AP (Access Point) and the entire region has an RP (Rendezvous Point). All members in a leaf cell of the hierarchy form a subgroup and are managed by that cell's AP. Groups of APs are managed recursively. Finally, APs belonging to the highest level of the hierarchy are managed by the RP. In this case, the area is divided in d^2 cells, each with one AP, and the d^2 APs are managed by the RP. The parameter d is called decomposition index.

To avoid the need of keeping track of the AP/RP nodes, needed in both membership management and data dissemination, which would require an external location service, HRPM adopts the idea of geographic hashing to reduce the maintenance of AP/RP nodes at virtually no maintenance cost. The role of each AP as well as the RP is mapped to a unique geographic location, via some simple hash function, the node that is currently closest to that location then serves the role of AP/RP, and routing to the AP/RP is conveniently achieved via geographic forwarding. There are rare cases in which messages sent to the RP/AP from different nodes may not converge to a single node. To solve this problem, when a node A receives the first packet from another node, and it thinks it is the RP/AP, it buffers the packet and starts an expanding ring broadcast search for any other node in the neighborhood, which also thinks it is the RP/AP. If such a node is found, A relays the buffered packet to that node. When a source has data packets to send, it first hashes the multicast group's identifier to obtain the location of the RP. It then contacts the RP and obtains the group membership vector, which specifies which cells have member. After that, the source builds an overlay tree, the Source → APs tree, considering each active AP as a vertex in a topology graph, and it sends data packets along the branches of this tree, using geographic forwarding. Each AP also builds an overlay tree, the AP → Members tree, whose vertices is the members in that AP's cell and forwards the data along this tree, also using geographic forwarding.

A minimum spanning tree (MST) achieves the best tradeoff among bandwidth efficiency, computational complexity, and location management overhead. Both the source and the APs use unicast to forward data packets, which means the same packet is sent over each branch of a sub tree, with a different header, specifying the path it should follow along that sub tree. Similarly to previously proposed geographic routingprotocols, HRPM has to deal with holes in the networktopology. Holes can occur in two different cases in HRPM,when routing to a node and when routing to a hashedlocation. The first case can happen when the AP →Members tree is traversed to deliver data to individualgroup members. This case is similar to geographic unicastrouting, and is handled using face-routing. Thesecond case can happen when the Source → APs tree istraversed to deliver data to APs. This case is slightly morecomplicated, since the node that encounters the hole has todistinguish whether the hole is en route to the hashedlocation or the hashed location is inside the hole. Similar tothe first case, the node starts face routing. If the packets traverse around the face and comes back to the node, thenthat node becomes the AP.

We now briefly present the procedure for calculating thedecomposition index d for the typical case of a two-levelhierarchy. Assume that the total number of group membersis G and the cost of encoding the node ID and its location isC bytes. Since the deployment area is divided into d^2 cells,the Source

P-3322

$\rightarrow$ APs tree has at most d^2 members, and theper-packet overhead is $d^2/8/f$ bytes, where f is the averagefan-out of the overlay tree at the root. Each AP$\rightarrow$Memberstree has on average G/d^2 members, and thus, the per-packetencoding overhead is at most $d^2=f$ bytes. d is calculatedbased on two constraints. The first constraint requires thatthe worst case encoding overhead in the AP $\rightarrow$ Memberstree be less than x bytes. With a worst-case fan-out fromthe tree root equal to 1. The second constraint requires that the worst caseencoding overhead in the Source $\rightarrow$ APs tree also be lessthan x bytes, i.e.,The RP first evaluates to select a value for d that islarge enough to satisfy that constraint. It then checks if thisvalue satisfies. For example, with a multicast group ofsize 100, using, with x = 102.4 bytes (20% of a 512-byte packet), and C = 12 bytes, we obtain d C=3.42. As thevalue d = 4 satisfies, we divide the network into 16cells. Detailed simulations showed that the values ofd chosen based on the above analysis yield optimalperformance.When the group size changes, the RP may decide toadjust d in order to satisfy constraint. In that case, itmulticasts a NOTIFY message with the new value for d toall members, using the current hierarchy. Upon receivingsuch a message, each member generates the hashed locationfor its new AP, and it sends an update to it. The newAPs then send the aggregated membership to the RP.

2.5 Hierarchical Geographic Multicast Routing (HGMR) [5]

Hierarchical Geographic Multicast Protocol (HGMR) which seamlessly combinesthe scalability of HRPM with theforwarding efficiency of GMR. The integration of theunique features of the two protocols poses a few interestingchallenges. The solution to reducing the encoding overheadis constructing a hierarchy of subgroups, similar to HRPM.For the delivery of the data to each of the subgroups,however, we could use either HRPM's (unicast-based) orGMR's (broadcast-based) forwarding strategy. GMR'sstrategy has the highest gain when the multicast memberdensity is large; in that case, the benefit from broadcastinga packet instead of unicasting it to each member is maximized.For the Source→APs overlay tree, the AP densityis expected to be low (one AP per cell), hence the benefitfrom using broadcast-based forwarding is not expected tobe large. In addition, for large networks, some APs will befar from the source and the overlay paths to them willinclude many hops, reducing reliability of message delivery. Hence, using unicast-based forwarding for such long pathshas one more advantage, since unicast MAC protocolsusually incorporate a hop-by-hop reliability mechanism, in contrast tobroadcast.

On the other hand, within each cell, the densityof multicast members is expected to be large, while thenumber of hops to each of them small. In this case,broadcast-based forwarding is preferred, since it can offer asignificant reduction in the number of transmissions.Based on the above observations, we now proceed todescribe the new protocol. HGMR divides the multicastgroup into subgroups using the mobile geographic hashingidea proposed in HRPM: the deployment area is againdivided into a number of cells; in each cell there is an APresponsible for all members in that cell, and all APs aremanaged by an RP. Membership management in HGMR isvery simple and of almost zero cost, thanks to the staticversion of the nodes and the use of geographic hashing. Tojoin a hierarchically decomposed multicast group, a nodegenerates the hashed location for the RP and sends a JOINmessage to that location. After receiving the value ofdecomposition index d from the RP, the node invokes thehash function with d and its location, to obtain the hashedlocation of the AP of the cell it belongs to. It then sends anUPDATE message to the AP. This completes the joinprocess. To leave a group, a node sends a LEAVE messageto the AP of the cell it belongs to. If that node was the onlymember the group in the cell, the AP has to notify the RP.Note that, in contrast to HRPM, no LOCATION-UPDATEmessages are required in HGMR, and no handoff process,since nodes are static and they do not move to differentcells.When a source has data packets to send, it uses HRPM'sunicast-based forwarding strategy to send the packets toeach AP along the Source→APs overlay tree.

But withineach cell, instead of constructing an AP→Membersoverlay tree, HGMR uses GMR's cost over progress optimizingbroadcast algorithm to select the next relay nodes ateach hop. Adjusting the value for the decomposition indexd, we can always ensure that the number of members an APis responsible for does not grow too large. Hence, the useof GMR within each cell instead of HRPM's unicast-basedforwarding strategy helps to reduce the number oftransmissions while maintaining a low encoding overhead.HGMR adopts both HRPM's and GMR's policies indealing with holes in sparse topologies. When routing to ahashed location (RP or AP), HGMR uses HRPM's facerouting, while when routing from an AP to a set of groupmembers within a cell, it uses GMR's multicast facerouting. Note, however, that holes are expected to be a veryrare case in HGMR which targets dense topologies.Thesetrees are constructed using GMR's localized neighborselection algorithm, and hence they are not overlay trees, asopposed to HRPM.

3. METHODOLOGY

3.1 Service Delivery [6]

Deliver relevant service data and control messages. Multicasting is used to support efficient group communication. Position based routing is, routing to the destination based on a nodes location. It's also known as geocasting. In geographic unicast routing mobile nodes are aware of their own positions and source can identify destination by location service, but they had high control overhead in dynamic environment. When service delivery is done by geographic multicast routing, the routing performance was increased but it needs information in packet header and is also hard to scale in large networks.

3.2 LeaderElection

Leaders are elected in each zone based on three algorithms.

- SGSP
- HRPM (Hash Function)
- Adaptive Core Node

3.3.1 Scalable Geographic Service Provisioning[6]

An access point (AP) is termed as the zone leader and electing it based on ZONE ID is one of the most conventional methods. In electing a global coordinator that is a rendezvous point (RP) the node with the highest zone ID is made the leader. When a node appears in the network it sends out a beacon announcing its existence. And then it waits for ainterval (min) period for the beacons from other nodes. Every interval (min) a node will check its neighbor table and decides its leader under different cases:

1) The neighbor table contains no other zNodes and it will announce itself as the leader.

2) All the zNodes flags are unset that means no zNode has announced the leadership role. If the node is closer to the zone centre than other nodes it will announce its leadership role through beacon message.

3) More than one zNodes have their flags set, the one with the largest node ID is selected. If the node's own flag is set before the checking but another node wins as zLD, the node will deliver its multicast table to the elected zone leader.

4) Just one flag is set for its zNodes with flagset is zLD.

So in simple terms for electing a zone leader that is an access point the node, with the highest node ID is selected.

3.3.2Hierarchical Rendezvous Point Multicasting[4]

The concept of group management is introduced assuming a flat geographic domain. We then introduce hierarchical domain decomposition of a multicast group and describe how to apply RPGM recursively in a hierarchy of sub domains. Rendezvous point group management allows multicast group members to leverage geographic hashing group management. Any node that wants to join a multicast group first hashes the group identifier o obtain the RP's location in the physical domain of the network using a hash function.

H(GID) ={x,y} where x,y € MANET region

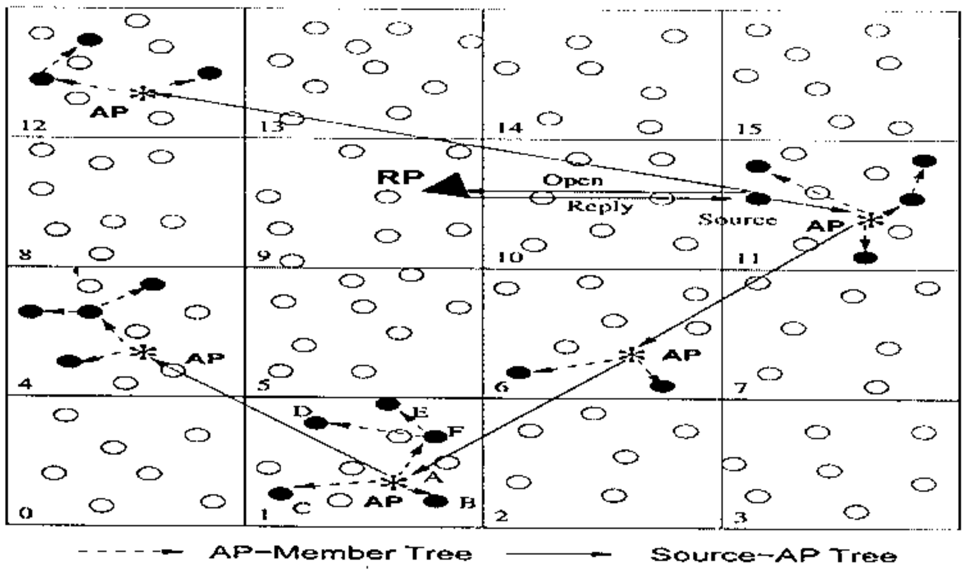


Fig 2: Data Delivery in HRPM

This hashing function takes as input the group identifier and outputs a location (x- and y- coordinates) contained in the region. Note that we assume that this is a well known hash function that is known by nodes that enter the network through external means or using some resource discovery process.

After obtaining the hashed RP location for the group it wants to join, the node sends a JOIN message addressed to this hashed location. This JOIN message is routed by geographic forwarding to the node that is current closest to the hashed location in the network. This node is the designated RP at this time. Since there is only one such node at any given time, the JOIN messages from all the group members converge at a single RP in a distributed fashion without global knowledge.

Note that computing the hashed location assumes that all nodes know the approximate geographic boundaries of the network. Such boundary information may be pre-configured at nodes before deployment or discovered using some simple protocol.

To join a hierarchically decomposed multicast group a node first generates the hashed location for the RP and sends a JOIN message to the RP, same as in a flat domain scenario. After receiving the value of the current decomposition index d of the hierarchy from the RP, the joining node invokes the hashing function with d and its current location to compute the hashed location of the AP of its cell. The node then starts periodically sending LOCATION UPDATE packets to its AP. Such location updates are soft state and serve as a subgroup membership update i.e. if an AP stops receiving location update from member; it assumes the member has migrated to another cell.

Upon receiving a location update from each member, the AP summarizes the membership inside its cell as non empty and further propagates to the RP whenever the membership switches between empty and non empty. The cells in which no group membersexist do not have any active APs and consequently no updates from these cells are sent to the RP.

3.3.3 Adaptive Core Node

Initially individual mobile users are ranked based on the predefined hierarchy and geographic domain of the organization. Based on the ranking mobile users are placed at respected levels. The top hierarchy is initially the global coordinator.

The global coordinator in turn delivers multicasting contents to the local coordinator and sometimes routed through zonal coordinators at their intermediate level. The local coordinator delivers multicasting content to the service nodes in the respective geographic region.

The idea of the voting algorithm is fetched from the cluster head election in clustering algorithm. In this, different parameters such as computational ability, battery power, the node ratio, transmission power, node mobility and node positioning are considered. The eligibility factor called EF of the node I that is utilized to serve as a CH at a particular time t is calculated as,

$$EF_i(t) = a_1 e^{-v_i(t)} + a_2 Bi(t) + a_3(1-E_i(t))$$

Where vi(t) is the mobile nodes average speed at time t and Bi(t) is the remaining battery power in node I at time t. Ei(t) is the Euclidean distance of the node I to the cluster centre calculated at time t and a1, a2,a3 are the weighting factors that reflect the importance of each parameter. The node with has the highest value of EF will elect itself as the CH in the cluster.

The same idea is followed for the election of AP and RP in our case except that for reducing the complexity we consider only three factors namely battery power, average speed of the nodes and the Euclidean distance from the zone centre. Battery power should be high for a node to stay alive longer and provide service. The average speed of the node should be less for it to stay in a particular zone for a longer time without showing faster switches between zones. The Euclidean distance between the node and its zone centre [6] should be less to receive packets quicker since the packets forwarded to a zone are forwarded towards its centre. The node which moderately excels in all the three factors is made the leader.

The leader election happens from time to time at fixed intervals. The RP and AP are elected at the start of the simulation. The leader election then happens at regular intervals such as 0 seconds, 7th second and fourteenth seconds based on the three algorithms zone ID based, hash based and voting algorithm based respectively. The RP leader election happens at the 0 and 9 th seconds.

3.3 Service delivery in SGSP [6]

When a service requestor (SR) wants to request one or multiple services, it will send a Query message with service descriptions (including service IDS and parameters)to an appropriate service coordinator. When there is a LC in its zone, it issues the Query to its LC; otherwise, if the SR knows RC, it will issue Query to RC. When neither LC nor RC is known, SR will send Query to GC directly or through GC-zone. If having no information on GC, GCZone, RC or LC, SR can start an expanded ring search, which is actually a fully distributed searching as in some service discovery protocols.

When a LC receives a Query or it itself has service requests, if any requested services can be satisfied by some local SP's according to its record, the Query will be forwarded to one or more candidate SPs. on receiving the Query, if it can provide the service, the SP will send back a Hit message describing what if can provide to the LC or initial SR (which one to respond to is based on the policy and the service request).if the requested services could not be satisfied locally, the LC will follow the same searching procedure as above and resort to its upper layer service coordinators, and the last option is to look for Sps through expanded ring search. When a RC or GC receives a query, it will process the message similarly. if any of the requested services can be satisfied by the service zones recorded, the Query will be further forwarded to the selected candidate zones. Selection criteria such as QoS requirements, geographic closeness will be forwarded to its LC, which will forward the Query to appropriate SPs to check if the services can be supported. Without receiving any hit message, the initial SR or corresponding coordinator can retry by reissuing the Query or resorting to another SP candidate or upper coordinator.

We also use service cache to optimize the service discovery. Each node keeps a service cache. Whenever a node receives a Hit or REGISTER message from a SP, it will cache the SP's position and the service information. The information will be removed after caching for a period initial cache. During the service discovery, when a non-coordinator node receives a Query, it will look up its service cache for the requested services, and forward the Query to qualified SPs if available, instead of forwarding the Query to destined coordinator, if the requested services cannot be satisfied by its service records, it will try to search for non local SPs in its service cache, and if this also fails, it will forward its Query to its upper coordinator.

3.4 Service delivery in HRPM [4]

The data delivery mechanism depends on the nature of the tree and the location/member management scheme used. A physical tree can efficiently be encoded at the header of a data packet. Such data packets can be delivered via source routing, as the tree contains all the intermediate nodes. In case of an overlay multicast tree, based on the group/location management scheme, there can be two approaches to data delivery:1)if the locations of the group members are known only to the source of the multicast tree, the destinations and the locations of the group members need to be encoded in the packet header at the source.And,2)if every group member knows every other group member's location ,only the destinations are encoded in the packet header(since each intermediate overlay node can fill in the locations and decide how the packet could be forwarded).This reduces the per-packet encoding overhead. However, this requires intermediate overlay nodes in the tree to acquire such location information via other means, for example, updates directly from the destination nodes. Moreover, in case of an overlay multicast tree, as the tree members may not be within direct reach of each other, geographic forwarding is needed to deliver data package along the overlay links.

In this paper, we use a greedy geographic forwarding algorithm as the routing protocol. Each node periodically announces its IP address and the location to its one-hop (within the radio transmission range) neighbors by broadcasting BEACON packets. Each node maintains the IP and the location information of its neighbors. Each packet being routed contains the destination address in the IP header and the destination's location(x and y -coordinates) in an IP option header. To forward a packet, a node consults its neighbor enable and forward s the packet to its neighbor that is closest in the geographic destination. Note that the above greedy geographic forwarding can lead to a packet reaching a node that does not know any other node that is closer to the destination than itself. This indicates a hole in the geographical distribution of the nodes. Recovering from the holes can be achieved using face routing.

HRPM provides a framework for scalable group management in the location-based multicast, in which any tree construction algorithm of choice can be utilized based on the application metrics. For the performance study in this paper, we assume the use of a specific overlay tree construction algorithm that minimizes the bandwidth cost. The source of the multicast Uses geographic distances between the multicast group members as edge weights to build an overlay graph, and then, a minimum spanning tree of the overlay graph(that is ,an overlay tree)is built by using MST algorithms (for example, Prims or Kruskal's). We evaluate different tree construction algorithms and show that such an overlay MST makes the best trade-off between the bandwidth efficiency, computational cost, and location management overhead.

To send a data packet, the source sends an OPEN SESSION message to the RP and receives the membership group vector from the RP. The membership vector is of size d2 bits, with a "1"bit for each cell that contains any group members. This vector is cached by the source. The RP differentially updates (sending only the changes) the source whenever the RP receives a change in membership notification from an AP .Once the group vector is received the source can build a virtual overlay tree (the Src→AP tree) by assuming each active AP as a vertex in a topology graph. The tree is virtual, since the source does not need to know the actual AP node in each cell: it just needs to hash the GID in the AP's cell to put in a virtual vertex in the topology graph.

Multicast data packets are first delivered down the Src→AP tree. In this phase, the encoding overhead is the bitmap of d2 bits (denoting which APs have active group members) that is inserted into every data packet. When a data packet reaches an AP, the AP performs the following operations:

It forwards the data packet on the remainder of the Src→AP tree below it by reconstructing the remaining tree based on the bitmap that it receives.
It constructs an AP→Member overlay tree to distribute this packet to the members of the group (the data packet has a destination group ID) in its cell by using the member locations that it stores. The AP then encodes the list of group members and

their each data packet sent along that branch. On the average, the number of group members in a cell is G/d2, where G is the group size. The packet is then delivered to the node down the AP→Member tree, with each node recomputing a tree of the remaining destinations in the list. Note that the size of this multicast header reduces as the packets travel down the tree and the depth of the remaining multicast tree reduces. Data delivery in HRPM for a multicast group that only has group members in cells 1,4,6,11,and12set from the Rp, since only those cells contain group members and, consequently, active APs.It then constructs a virtual topology graph containing all the active APs, and builds a Src-->AP multicast tree containing the active Aps.The multicast data packet is then sent out on each branch(two at the source):one toward the Ap in cell 12 with bitmap[0000000000001000]and one toward the AP in cell 11 with bitmap[0100101000010000].

On receiving the packet,the AP in cell 12flips the bit that corresponds to its cell and finds no further APs,so it only sends the packet to the members in its cell.In contrast,when the AP in cell 11 receives the packets, it flips bit 11 and finds three remaining AP's in the bitmap. It then constructs a tree to these three APs, encodes a new bitmap (without its own bit) and sends the packet. To see how the AP→Member Tree works consider the AP in cell 1. It constructs a tree with group members (B-F) and splits the packet three ways in the first branch of the tree. Node F can see the remaining group members in its branch (E,D) when it receives the packet and thus reconstructs the tree to send the packet forward.

Multiple Sources

Multiple sources for a single group work similarly: each source retrieves from the RP and constructs its independent Source→AP tree. Each AP can cache and reuse the AP→Member tree (saves computation) for delivering packets for multiple sources to members in its cell.

Since the primary focus of this paper is on multicast routing and group management, we do not address reliability and security issues due to lack of space. As with all multicast protocols, the malicious operation of nodes or the failure of nodes can cause service disruptions. Mechanisms for dealing with these problems are part of our future work.

3.5 Adaptive Service Delivery

3.5.1 Disadvantage of HRPM in Service Delivery

- Service Requestor must run the hash function to get the core node location every time it needs a service.
- All the service providers gets the bit vector from rendezvous point periodically and deliver the service. This results in reduntant service delivery wasting network resources.
- Every group has an unique access point and rendezvous point. Therefore if a service provider provides n services it has to construct n spanning trees from service providers to access point and n cost over progress tree from access point to destinations

3.5.2 Tree Aggregation

In adaptive service delivery, service provider (SP) periodically delivers the service to all the service requestors (SR). A service provider may provide one or more services. Here the access point (AP) and rendezvous point (RP) are elected based on the adaptive core node election technique. The access point maintains a table which contains the list of all the service requestors in the particular zone. The serviceprovider maintains a table which contains the details of all the zones, nodes and the access pints. Both the access point and the service provider are updated periodically in order to update the location information of all the nodes.

When a particular service needs to be delivered to a service requestor first the service provider forms a bit vector which contains the details of all the service requestors' zone idand its access point. The service providercontains more number of services so it form separate bit vector for each services. Then these bit vector are aggregated into a single bit vector using AND operation.The bit vector contains the zones of the service requestors. The inter zone aggregated tree constructed is carried out using prims shortest path algorithm and intra zone aggregated tree is constructed using GMR [4]. Using the bit vector the services are delivered to the corresponding zone. The tree constructed using the prims shortest path algorithm is called Minimum Spanning Tree (MST).

The geographic multicast routing algorithm provides an efficient cost over progress ratio by efficiently choosing the relay node. A relay node is simply a node that provides maximum forwarding efficiency. The same node can act as a relay node for more than one service requestors. Once the relay node is selected then the service is delivered adaptively from the access point to the service requestors via the relay node.

3.5.3 Algorithm for MST Based Tree Construction

```
1: for each v 2 Child(u)
2: if TYPE = SPT
3: d[v] = d[u] + w(u, v)
4: if TYPE = MST
5: d[v] = w(u, v)
6: if TYPE = COM
7: d[v] = a * d[u] + w(u, v)
8: AdjustWeight(TYPE, v) /*Adjust Weight in
thesubtree */
```

```
1: /*Energy Minimizing Phase*/
2: Initialize(G,s)
3: S = ;
4: PQ = V[G]
5: while PQ 5 ;
6: u = Extract_min(PQ)
```

```
7: if (u 5 s)
8: p = p[u]
9: Child[p] = Child[p] [ {u}
10: S = S [ {u}
11: for all v 2 Adj[u]
12: Relax(TYPE,u, v) /* TYPE is SPT,
MST or COM (for Combined-SPT–MST) */
13:

14: /*Latency Minimizing Phase*/
15: AdjustTree(s)
16: AdjustWeight(TYPE, s) /* TYPE is SPT,
MST or COM (for Combined-SPT–MST) */
1: /*Energy Minimizing Phase*/
2: Initialize(G,s)
3: while PQ 5 ;
4: u = Extract_min(PQ)
5: if u 5 s
```

```
6: Child[p(u)] = Child[p(u)] [ {u}
7: for all v 2 Adj[u]
8: if v 2 PQ and w(u, v) < d[v]
9: Relax(MST,u, v)
10:
11: /*Latency Minimizing Phase*/
12: AdjustTree(s)
13: AdjustWeight(MST, s)
```

3.5.4 Advantages of Adaptive Service Delivery

- Service providers maintains the list of service requestors ID and location. Therefore it can construct a spanning tree periodically from service requestor to access point.

- During adaptive service delivery the service requestor chooses the service requestor chooses that is close to it. Therefore reduntant service delivery is prevented.

- In adaptive core node election mechanism, only one core node is choosen per zone for any number of groups. This results in very less number of trees.

4. SIMULATION SCENARIO

We have implemented the proposed scheme using NS2 on LINUX platform. NS simulator is based on two languages – an object oriented simulator written in C++ and an object oriented extension of Tcl which is used to execute user command scripts. It also has rich library of network and protocol objects.

We represent our scheme as ASDP (Adaptive Service Discovery Protocol). We have used AODV (Ad hoc On-demand Distance Vector) routing protocol which is inbuilt in NS2 in a hierarchical topology. AODV has the basic route-discovery and route-maintenance of DSR (Dynamic Source Routing) and uses the hop-by-hop routing, sequence numbers and beacons of DSDV (Destination Sequence Distance Vector). The node that wants to know a route to a given destination generates a ROUTE REQUEST. The route request is forwarded by intermediate nodes that also create a reverse route for themselves from the destination. When the request reaches a node with route to destination it generates a ROUTE REPLY containing the number of hops required to reach destination.

We have deployed Random Waypoint Mobility Model in which each mobile node chooses a random destination and moves towards it with a random velocity chosen from [0, Vmax], where Vmax is the maximum speed of the mobile node. After reaching the destination, the node stops for a duration defined by the "pause time" parameter. After this duration, it again chooses a random destination and repeats the whole process again until the simulation ends.

We have included the energy model which represents level of energy of a mobile node. The energy model in a node has an initial value which is the level of energy the node has at the beginning of the simulation. This is known as initialEnergy_. This parameter is set to 0.5 Joules. There is also energy usage for every packet a node transmits and receives. These are called txPower_ and rxPower_. The values of these parameters are set to 0.3 and 0.6 Watt.

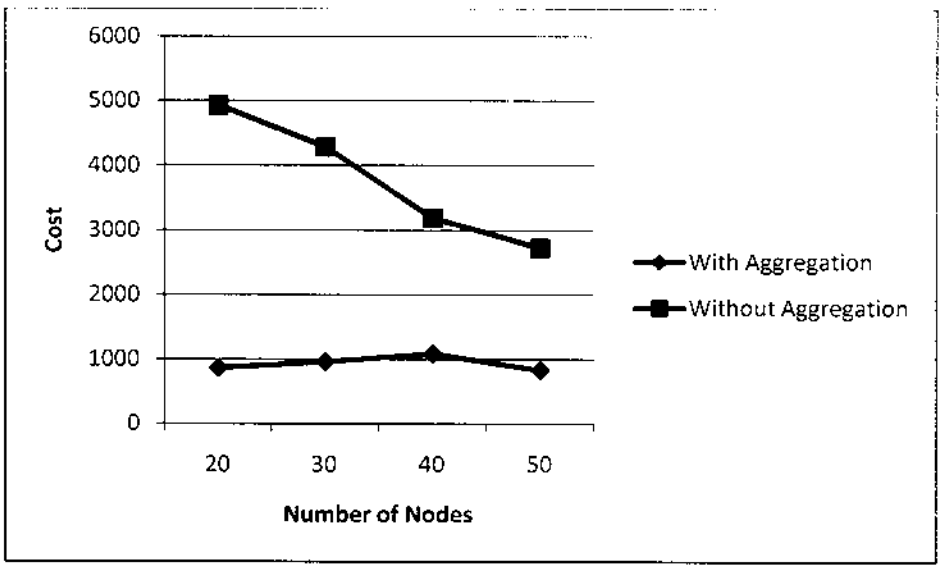
We have assumed the MANET region as a square region of size 2400mX2400m. The simulations were run about 600 seconds for 30, 40 and 50 nodes for Vmax values – 10, 25, 50, 75, 100, 125, 150, 175, 200 m/s and pause time 2, 5, 10 seconds. During virtual zone construction, the zone-size is taken as 800m which resulted in 9 different zones with ZIDs (0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2). The nodes were randomly distributed by the Random Waypoint Model.

In order to facilitate performance measurement, we have fixed the acceptable load for service types as 75% of the total number of nodes.

5. PERFORMANCE EVALUATION

5.1 Effect Of Mobility On Total Cost

Cost is the total distance travelled by a packet from service provider to service requestor.

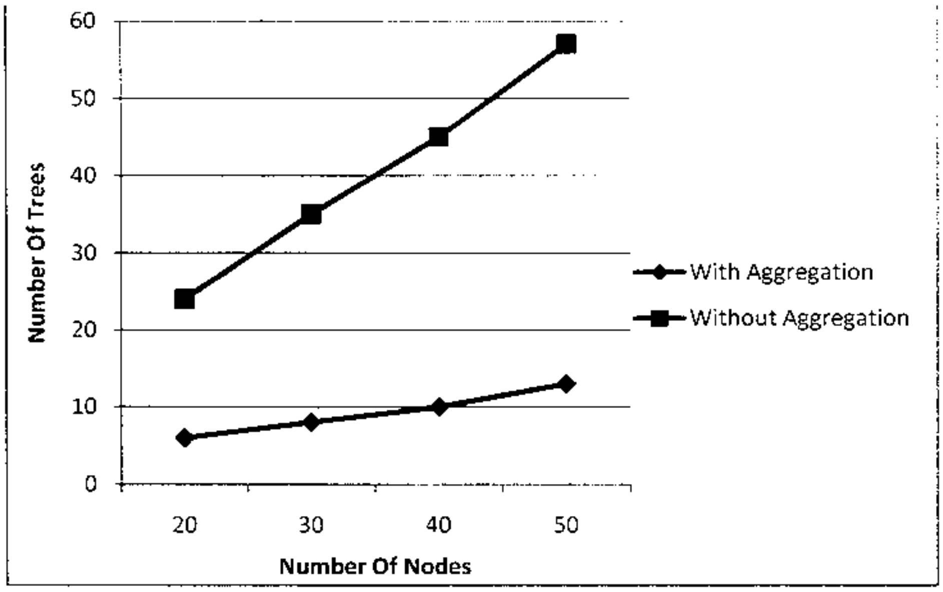


Inference

The distance travelled by a packet when using aggregation technique is much less when compared to the distance travelled by the packet without using aggregation technique.

5.2 Effect Of Mobility On Number Of Spanning Trees

One service provider may provide more than one service. In adaptive service delivery we aggregate the spanning trees of all the services from a particular SP.

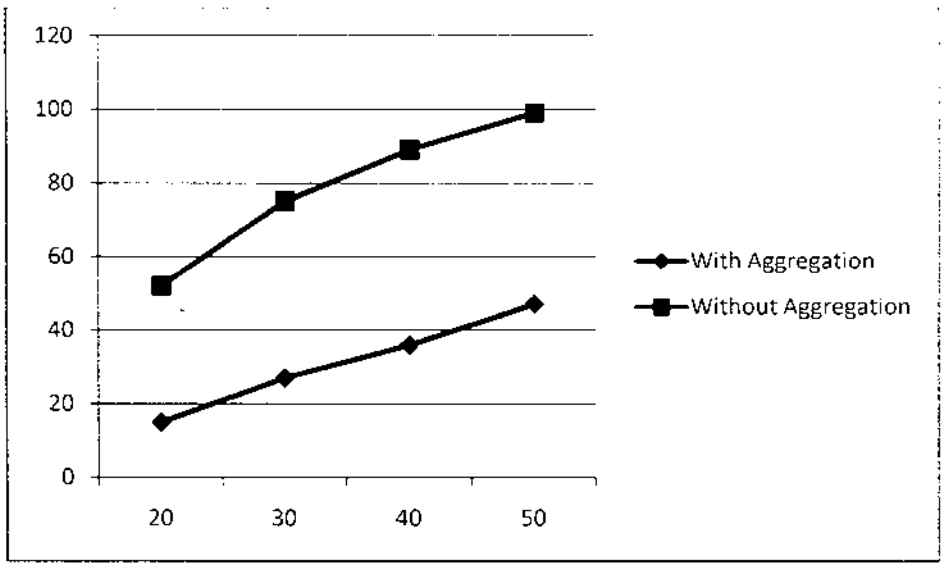


Inference

The above analysis shows that the number of spanning trees using aggregation technique is very low when compared number of spanning trees without using aggregation technique. Reduction of number of trees in turn reduces the wastage of network resources.

5.3 Effect Of Mobility On Cost Over Progress Tree

The cost over progress tree can beused to select the next hops towards destinations.



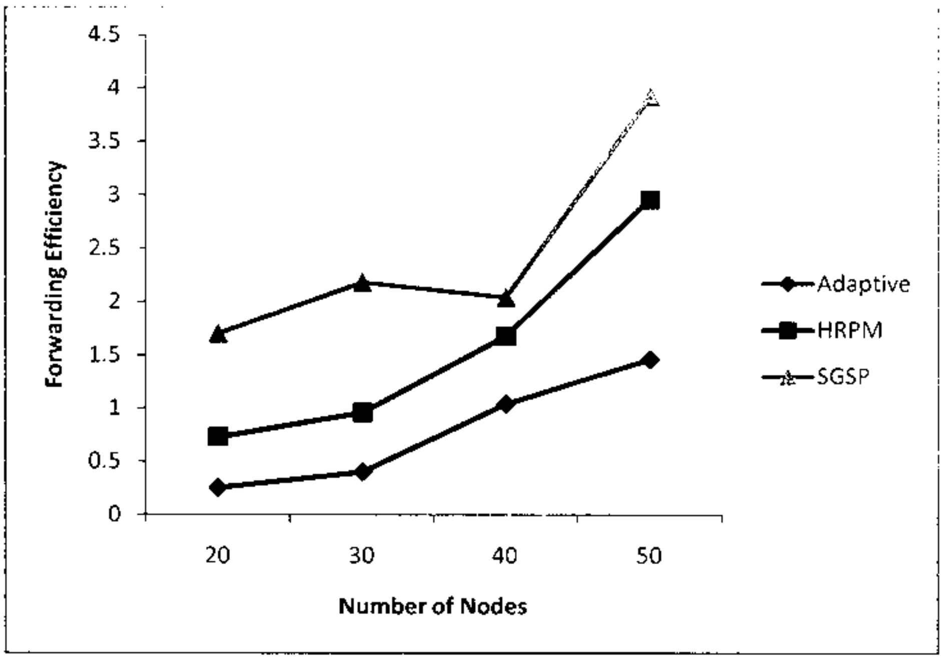
Inference

The above analysis shows that the number of cost over progresstrees using aggregation technique is very low when compared number of cost over progresstrees without using aggregation technique. Reduction of number of trees is due to the appropriate selection of the relay nodes.

5.4 Forwarding Efficiency

Forwarding nodes are the nodes that forward atleast one packet.

Forwarding Node Efficiency=No:of forwarding nodes/No:of nodes

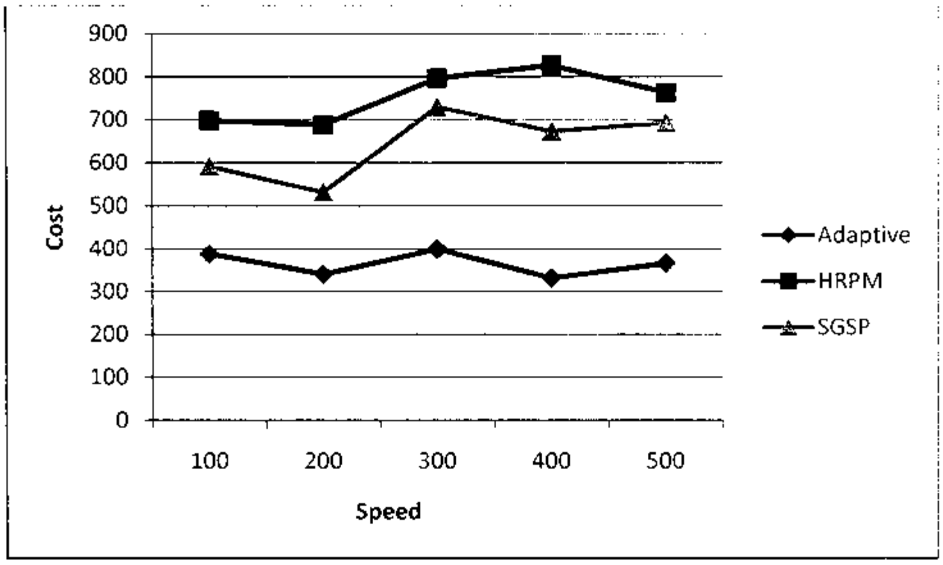


Inference

The forwarding efficiency is greater for adaptive core node technique than HRPM and SGSP. This is due to the aggregation of the delivery trees.

5.5 Effect Of Speed

The distance between the nodes varies as the speed of the node varies.

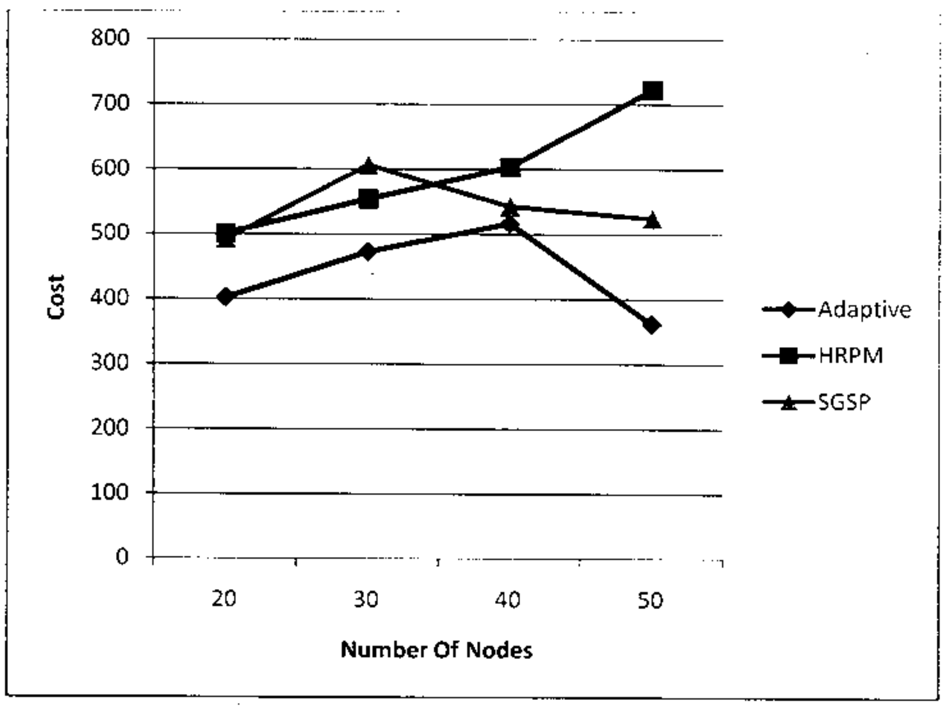


Inference

This proves that adaptive core node technique cost is less than HRPM and SGSP technique when the speed of the nodes gets increased. This shows that adaptive core node technique is more efficient than HRPM and SGSP

5.6 Effect Of Number Of Nodes

The cost of a node varies as the number of node in a given network varies.

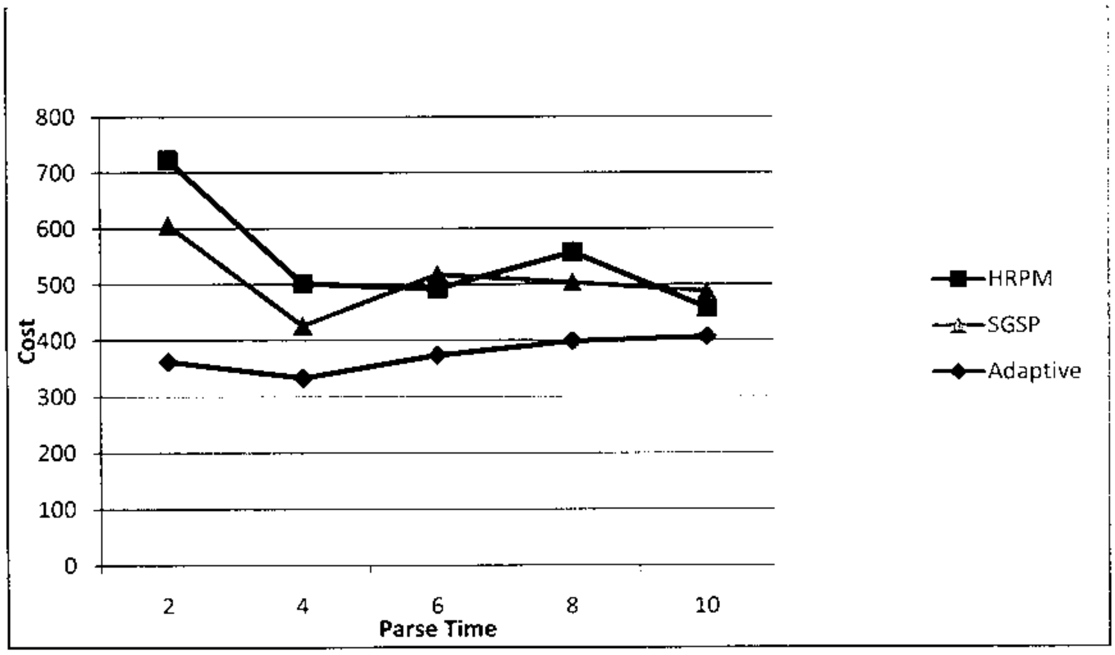


Inference

This proves that the cost is very low as the number of nodes gets increased in adaptive core node technique as compared with HRPM and SGSP. Hence adaptive core node technique is very cost efficient.

5.7 Effect Of Pause Time

Pause time is defined as the time in which a node stays at a particular location in a mobile adhoc network.



Inference

This proves that parse time is very low in adaptive core node technique when compared with HRPM and SGSP technique.

CONCLUSION

Wireless networks are energy constrained network. Since most of the energy consumed for transmitting and receiving data, the process of tree aggregation becomes an important issue and optimization is needed. Efficient tree aggregation protocols not only provide energy conservation but also remove redundancy in the data and hence provide useful data only. There exist several protocols for tree aggregation which uses different approaches to provide energy efficiency. The distance travelled by a packet is largely reduced and the efficiency increases as the number of packets and the number of destination increases. The throughput is also better when compared to other multicasting scenarios.

FUTURE ENHANCEMENT

The system is more scalable in terms of number of multicast groups compared to other protocols. Still the future research can be made in the following direction.

- Intend to consider the reconstruction of the tree if a hole is discovered by applying effective recovery strategy.
- Intend to apply data aggregation at each hierarchical level to reduce traffic in the network.

APPENICES

APPENDIX 1 – SOURCE CODE

```
proc tree { timea } {
        global z1 z2 z3 z4 z5 z6 z7 z8 z9
        global r1 r2 r3 r4 r5 r6 r7 r8 r9
        global zone leader loc
        global ns_ node_
        for {set i 0} {$i<$r1} {incr i} {
                set zone(0,$i) $z1($i)
                set loc($z1($i)) $leader(0)
        }
        for {set i 0} {$i<$r2} {incr i} {
                set zone(1,$i) $z2($i)
                set loc($z2($i)) $leader(1)
        }
        for {set i 0} {$i<$r3} {incr i} {
                set zone(2,$i) $z3($i)
                set loc($z3($i)) $leader(2)
        }
        for {set i 0} {$i<$r4} {incr i} {
                set zone(3,$i) $z4($i)
                set loc($z4($i)) $leader(3)
        }
        for {set i 0} {$i<$r5} {incr i} {
                set zone(4,$i) $z5($i)
                set loc($z5($i)) $leader(4)
        }
        for {set i 0} {$i<$r6} {incr i} {
                set zone(5,$i) $z6($i)
                set loc($z6($i)) $leader(5)
        }
        for {set i 0} {$i<$r7} {incr i} {
                set zone(6,$i) $z7($i)
                set loc($z7($i)) $leader(6)
        }
        for {set i 0} {$i<$r8} {incr i} {
                set zone(7,$i) $z8($i)
                set loc($z8($i)) $leader(7)
        }
        for {set i 0} {$i<$r9} {incr i} {
                set zone(8,$i) $z9($i)
                set loc($z9($i)) $leader(8)
}
        for { set e 0} {$e < 40} {incr e} {
                set x($e) "[ $node_($e) set X_ ]"
                set y($e) "[ $node_($e) set Y_ ]"
        }
        global ns_ node_ leader
        set count 0
        for {set u 0} {$u<40} {incr u} {
        for {set w 0} {$w<"[ $node_($u) set service_count ]"} {incr w} {
                for {set i 0} {$i<40} {incr i} {
                        set vv " [ $node_($i) set service_count ] "
                for { set j 0 } { $j < $vv } {incr j } {
                        global services
                        if { "[$node_($u) set services($w)]" == "[ $node_($i) set
services($j) ]" } {
                        set ser_pro($count) $i
                        incr count
                        }
                }
        set tem [expr $i + 3 ]
        set i $tem
        }
        for {set i 0} {$i<3} {incr i} {
        set ap $loc([$node_($u) set ser_req($w,$i)])
                for {set j 0} {$j<$count} {incr j} {
                set dx [expr $x($ap) - $x($ser_pro($j))]
                set dy [expr $y($ap) - $y($ser_pro($j))]
                set dist [expr ($dx*$dx) - ($dy*$dy)]
                if { $dist<0 } {
                        set dist [expr (-1)*$dist]
                }
                set distance [exprsqrt ($dist)]
                set finaldis($i,$j) $distance
                set finaldis1($i,$j) $distance
                }
                if {$count>1} {
                for {set k 0} {$k<$count} {incr k} {
                        for {set j [expr $k + 1]} {$j< $count} {incr j} {
```

```
                                        if { $finaldis($i,$k) > $finaldis($i,$j) } {
                                                set t $finaldis($i,$k)
                                                set finaldis($i,$k) $finaldis($i,$j)
                                                set finaldis($i,$j) $t
                                        }
                                }
                        }
                }
                for {set j 0} {$j<$count} {incr j} {
                        if {$finaldis($i,0)==$finaldis1($i,$j)} {
                                set q "[$node_($u) set ser_req($w,$i)]"
                                set p  $loc($q)
                                for {set k 0} {$k<9} {incr k} {
                                        if {$p==$leader($k)} {
                                                set v $k
                                        }
                                }
                                $node_($ser_pro($j)) set ser_bv($w,$v) 1
                        }
        }
        }
        set count 0
    }
    set u [expr $u + 3]
    }
set viji 0
for {set u 0} {$u<40} {incr u} {
        puts "for SP node_($u)"
        incrviji
        puts "zone          |      0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |
        8  |"
        puts "---------------------------------------------------------------------------
----------"
        for {set w 0} {$w<"[ $node_($u) set service_count ]"} {incr w} {
                puts "service  [$node_($u) set services($w)]         [$node_($u) set
ser_bv($w,0)]       [$node_($u) set ser_bv($w,1)]  [$node_($u) set ser_bv($w,2)]
        [$node_($u) set ser_bv($w,3)]  [$node_($u) set ser_bv($w,4)]  [$node_($u)
set ser_bv($w,5)]  [$node_($u) set ser_bv($w,6)]  [$node_($u) set ser_bv($w,7)]
        [$node_($u) set ser_bv($w,8)]"
                for {set i 0} {$i<9} {incr i} {
                        if {"[$node_($u) set ser_bv($w,$i)]"==1} {
                                $node_($u) set fin($i) 1
                        }
                }
        }
        puts "---------------------------------------------------------------------------
----------"
        puts "Result                   [$node_($u) set fin(0)]  [$node_($u) set fin(1)]
        [$node_($u) set fin(2)]   [$node_($u) set fin(3)]   [$node_($u) set fin(4)]
        [$node_($u) set fin(5)]   [$node_($u) set fin(6)]   [$node_($u) set fin(7)]
        [$node_($u) set fin(8)]"
        puts "---------------------------------------------------------------------------
----------"
        puts ""
        set scr1 $u
        set cnt 0
        for {set i 0} {$i<9} {incr i} {
                if {"[$node_($u) set fin($i)]"==1} {
                set dest($cnt) $i
                puts "AP of zone $i: $leader($i)"
                incrcnt
                }
        }
        set counter $cnt
        set p $loc($u)
        for {set i 0} {$i<9} {incr i} {
                if {$p==$leader($i)} {
                set v $i
                }
        }
        set first $v
        set fla 0
        for {set i 0} {$i<$cnt} {incr i} {
                if {$dest($i)==$first} {
                set fla 1
                }
    }
                        if {$fla==0} {
                        set dest($cnt) $first
                        incrcnt
                        }
                for {set i 0} {$i<$cnt} {incr i} {
                 for {set j 0} {$j<$cnt} {incr j} {
                 set otree($i,$j) 0
                 }
                 }
set pathdis 0
for {set i 0} {$i<$cnt} {incr i} {
for {set j 0} {$j<$cnt} {incr j} {
set dis($i,$j) 0
}
}
for {set i 0} {$i<$cnt} {incr i} {
for {set j 0} {$j<$cnt} {incr j} {
if {$i!=$j} {
        set g $dest($i)
        set v $dest($j)
       if {$x($leader($g))>$x($leader($v))} {
           set dx [expr $x($leader($g))-$x($leader($v))]
            }
      if {$x($leader($g))<$x($leader($v))} {
           set dx [expr $x($leader($v))-$x($leader($g))]
           }
if {$y($leader($g))>$y($leader($v))} {
                   set dy [expr $y($leader($g))-$y($leader($v))]
              }
                 if {$y($leader($g))<$y($leader($v))} {
                     set dy [expr $y($leader($v))-$y($leader($g))]
                }
                set q [expr [expr $dx*$dx]+[expr $dy*$dy]]
             set dis($i,$j) [exprsqrt($q)]
        }
        }
}
#Constructing tree using prim's algorithm
puts "The tree connecting the APs of the destination zones"
        puts "--------------------------------------------------"
        set f [expr $cnt-1]
        for {set i 0} {$i<$cnt} {incr i} {
                for {set j 0} {$j<$cnt} {incr j} {
                        if {$dis($i,$j)==0} {
                                set dis($i,$j) 9999
                        }
                }
        }
        for {set m 0} {$m<$cnt} {incr m} {
                set s($m) 0
        }
        for {set m 0} {$m<$cnt} {incr m} {
                if {$dest($m)==$first} {
                        set s($m) 1
                }
        }
        set ap_cnt 0
        for {set ne 0} {$ne<$f} {incr ne} {
```

```
                    set min 9999
                    for {set i 0} {$i<$cnt} {incr i} {
                            if {$s($i)==1} {
                                    for {set j 0} {$j<$cnt} {incr j} {
                        ,   if {$s($j)==0} {
                                            if {$min>$dis($i,$j)} {
                                                    set min $dis($i,$j)
                                                    set x2 $i
                                                    set y2 $j
                                            }
                                    }
                            }
                            }
                    }
                    set s($y2) 1
                    set ww [expr $x2+1]
                    set q [expr $y2+1]
                    set pdis $pathdis
                    set pathdis [expr $pdis+$dis($x2,$y2)]
                    if {$dest($x2)==$first} {
                            puts "S.P $scr1 --->A.P $leader($dest($y2))"
                            set ap_arr($ap_cnt) $leader($dest($y2))
                            incrap_cnt
                    } else {
                            puts "A.P $leader($dest($x2)) --->A.P
$leader($dest($y2))"
                            set ap_arr($ap_cnt) $leader($dest($y2))
```

```
        incrap_cnt
    }
    set otree($x2,$y2) 1
    set xe [expr $ne+1]
    #set up a tcp connection between node x and node y
    set tcp_(0) [new Agent/TCP/Newreno]
    $tcp_(0) set class_ 2
    set sink_(0) [new Agent/TCPSink]
    $ns_ attach-agent $node_($x2) $tcp_(0)
    $ns_ attach-agent $node_($y2) $sink_(0)
    $ns_ connect $tcp_(0) $sink_(0)
    set ftp_(0) [new Application/FTP]
    $ftp_(0) attach-agent $tcp_(0)
    set udp_($x2) [new Agent/UDP]
    $ns_ attach-agent $node_($x2) $udp_($x2)
    set null_($x2) [new Agent/Null]
    $ns_ attach-agent $node_($y2) $null_($x2)
    set cbr_($x2) [new Application/Traffic/CBR]
   $cbr_($x2) set packetSize_ 512
   $cbr_($x2) set interval_ 4.0
   $cbr_($x2) set random_ 1
   $cbr_($x2) set maxpkts_ 10000
    $cbr_($x2) attach-agent $udp_($x2)
    $ns_ connect $udp_($x2) $null_($x2)
    set xc [expr $timea + 0.2]
    $ns_ at $xc "$ftp_(0) start"
    $ns_ at $xc "$cbr_($x2) start"
```

```
                    $ns_ at $xc "$cbr_($x2) stop"
                    $ns_ at $xc "$ns_ trace-annotate \"Packet transmitted from AP
$leader($dest($x2)) to AP $leader($dest($y2)) \""
                    $ns_ at $xc "$ns_ trace-annotate \" Distancehrpm $dis($x2,$y2)
\""
                    $ns_ at $xc "$ns_ trace-annotate \" Viji $viji \""
                    set scr2 $leader($dest($y2))
                    if {$dest($y2)==0} {
                            set r $r1
                            set iii 0
                    }
            if {$dest($y2)==1} {
                    set r $r2
                    set iii 1
                    }
                    if {$dest($y2)==2} {
                            set r $r3
                            set iii 2
                    }
            if {$dest($y2)==3} {
                            set r $r4
                            set iii 3
                    }
            if {$dest($y2)==4} {
                            set r $r5
                            set iii 4
                    }
                    if {$dest($y2)==5} {
                            set r $r6
                            set iii 5
                    }
                    if {$dest($y2)==6} {
                            set r $r7
                            set iii 6
                            }
                    if {$dest($y2)==7} {
                            set r $r8
                            set iii 7
                    }
                    if {$dest($y2)==8} {
                            set r $r9
                            set iii 8
                    }
                    set node_cnt 1
                    for {set i 0} {$i<"[ $node_($u) set service_count ]"} {incr i} {
                            for {set j 0} {$j<3} {incr j} {
                                    for {set k 0} {$k<$r} {incr k} {
                                            if {"[$node_($u) set
ser_req($i,$j)]"==$zone($dest($y2),$k)} {
                                                    set dest2($node_cnt)
$zone($dest($y2),$k)
                                                    incrnode_cnt
                                            }
                                    }
```

```
        }
}
set x1 $node_cnt
set cnt1 $node_cnt
set dest2(0) $scr2
for {set i 0} {$i<$cnt1} {incr i} {
        for {set j 0} {$j<$cnt1} {incr j} {
            set dis1($i,$j) 0
        }
}
for {set i 0} {$i<$cnt1} {incr i} {
        for {set j 0} {$j<$cnt1} {incr j} {
                if {$i!=$j} {
                        set z $dest2($i)
                      set v $dest2($j)
                        if {$x($z)>$x($v)} {
                          set dx [expr $x($z)-$x($v)]
                      }
                if {$x($z)<$x($v)} {
                          set dx [expr $x($v)-$x($z)]
                    }
                      if {$y($z)>$y($v)} {
                            set dy [expr $y($z)-$y($v)]
                      }
                if {$y($z)<$y($v)} {
                          set dy [expr $y($v)-$y($z)]
                      }
                set q [expr [expr $dx*$dx]+[expr $dy*$dy]]
            set dis1($i,$j) [exprsqrt($q)]
        }
    }
}
set zone_nodecnt 0
set flag_nodecnt 0
for {set i 0} {$i<$r} {incr i} {
        for {set j 0} {$j<$cnt1} {incr j} {
                if {$zone($iii,$i)==$dest2($j)} {
                        set flag_nodecnt 0
                        break
                } else {
                        set flag_nodecnt 1
                }
        }
        if {$flag_nodecnt==1} {
                set zone_node($zone_nodecnt) $zone($iii,$i)
                #puts "$zone_node($zone_nodecnt)"
                incrzone_nodecnt
        }
        set flag_nodecnt 0
}
for {set i 0} {$i<$cnt1} {incr i} {
        set relay($dest2($i)) -1
}
for {set i 0} {$i<$zone_nodecnt} {incr i} {
```

```
                    set temp_cnt 0
                    for {set j 1} {$j<$cnt1} {incr j} {
                         set dx [expr [$node_($dest2($j)) set X_]-
[$node_($zone_node($i)) set X_] ]
                         set dy [expr [$node_($dest2($j)) set Y_]-
[$node_($zone_node($i)) set Y_] ]
                         set dd [expr ($dx*$dx) + ($dy*$dy)]
                         if {$dd<1} {
                              set dis_dd [expr (-1) * $dd]
                              set dist_dd [exprsqrt($dis_dd)]
                         } else {
                              set dist_dd [exprsqrt($dd)]
                         }
                         if {$dist_dd<$dis1(0,$j)} {
                              if {$dest2($j)==$dest2(0)} {
                                   set relay($dest2($j)) -1
                              } else {
                                   set relay($dest2($j)) $zone_node($i)
                                   #set rej($rej_cnt) $j
                                   #puts "j:$j"
                                   continue
                              }
                         } else {
                              set relay($dest2($j)) -1
                         }
                    }
               }
```

```
                puts "Tree Construction within zone"
                puts "Relay nodes are"
                for {set i 1} {$i<$node_cnt} {incr i} {
                        if {$dest2($i)==$dest2(0)} {
                        } else {
                                if {$relay($dest2($i))==-1} {
                                } else {
                                        puts "relay($dest2($i)) :$relay($dest2($i))"
                                }
                        }
                }
                puts "tree"
                for {set i 1} {$i<$node_cnt} {incr i} {
                        if {$dest2($i)==$dest2(0)} {
                        } else {
                                if {$relay($dest2($i))==-1} {
                                        puts "$dest2(0)---->$dest2($i)"
                                        set dx [expr [$node_($dest2(0)) set X_]-
[$node_($dest2($i)) set X_] ]
                                        set dy [expr [$node_($dest2(0)) set Y_]-
[$node_($dest2($i)) set Y_] ]
                                        set dd [expr ($dx*$dx) + ($dy*$dy)]
                                        if {$dd<1} {
                                                set dis_dd [expr (-1) * $dd]
                                                set dist_dd [exprsqrt($dis_dd)]
                                        } else {
                                                set dist_dd [exprsqrt($dd)]
```

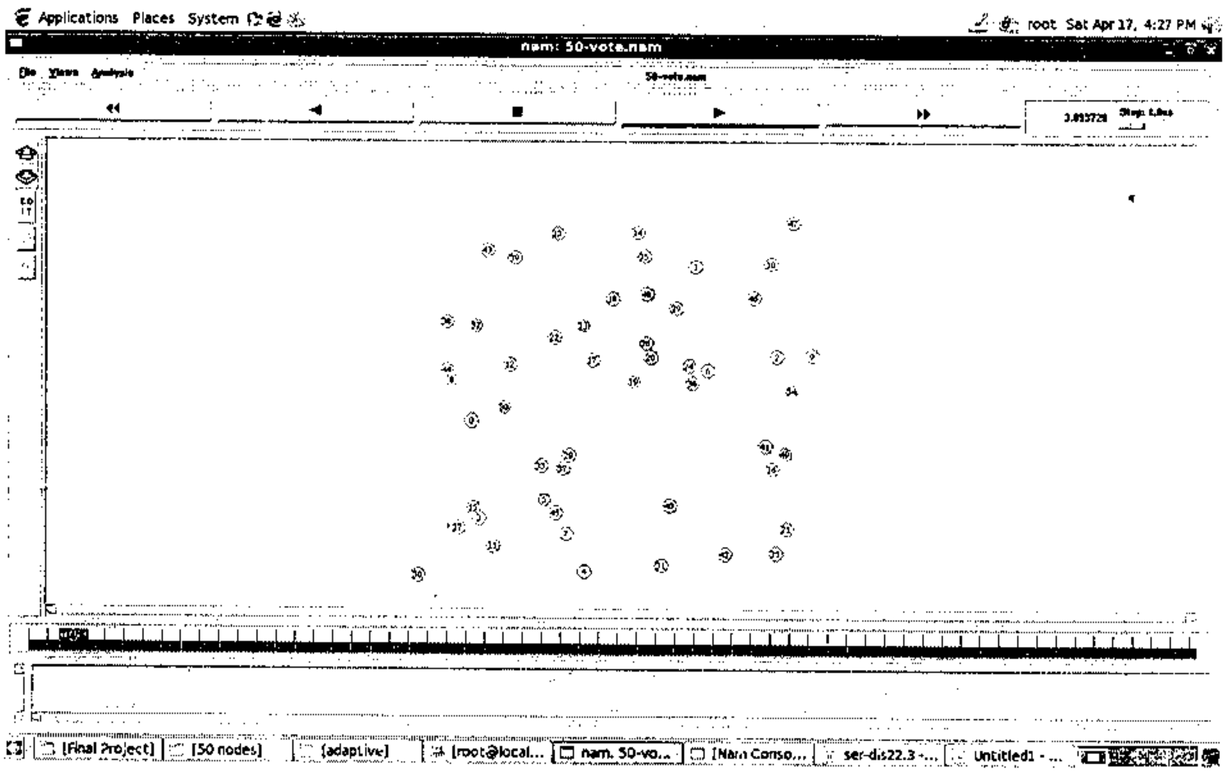
```
                                   }
                                   set tcp_(0) [new Agent/TCP/Newreno]
                                   $tcp_(0) set class_ 2
                                   set sink_(0) [new Agent/TCPSink]
                                   $ns_ attach-agent $node_($dest2(0))
        $tcp_(0)
                                   $ns_ attach-agent $node_($dest2($i))
        $sink_(0)
                                   $ns_ connect $tcp_(0) $sink_(0)
                                   set ftp_(0) [new Application/FTP]
                                   $ftp_(0) attach-agent $tcp_(0)
                                   set udp_($dest2(0)) [new Agent/UDP]
                                   $ns_ attach-agent $node_($dest2(0))
        $udp_($dest2(0))
                                   set null_($dest2(0)) [new Agent/Null]
                                   $ns_ attach-agent $node_($dest2($i))
        $null_($dest2(0))
                                   set cbr_($dest2(0)) [new
        Application/Traffic/CBR]
                                  $cbr_($dest2(0)) set packetSize_ 512
                                  $cbr_($dest2(0)) set interval_ 4.0
                                  $cbr_($dest2(0)) set random_ 1
                                  $cbr_($dest2(0)) set maxpkts_ 10000
                                   $cbr_($dest2(0)) attach-agent
        $udp_($dest2(0))
                                   $ns_ connect $udp_($dest2(0))
        $null_($dest2(0))
```

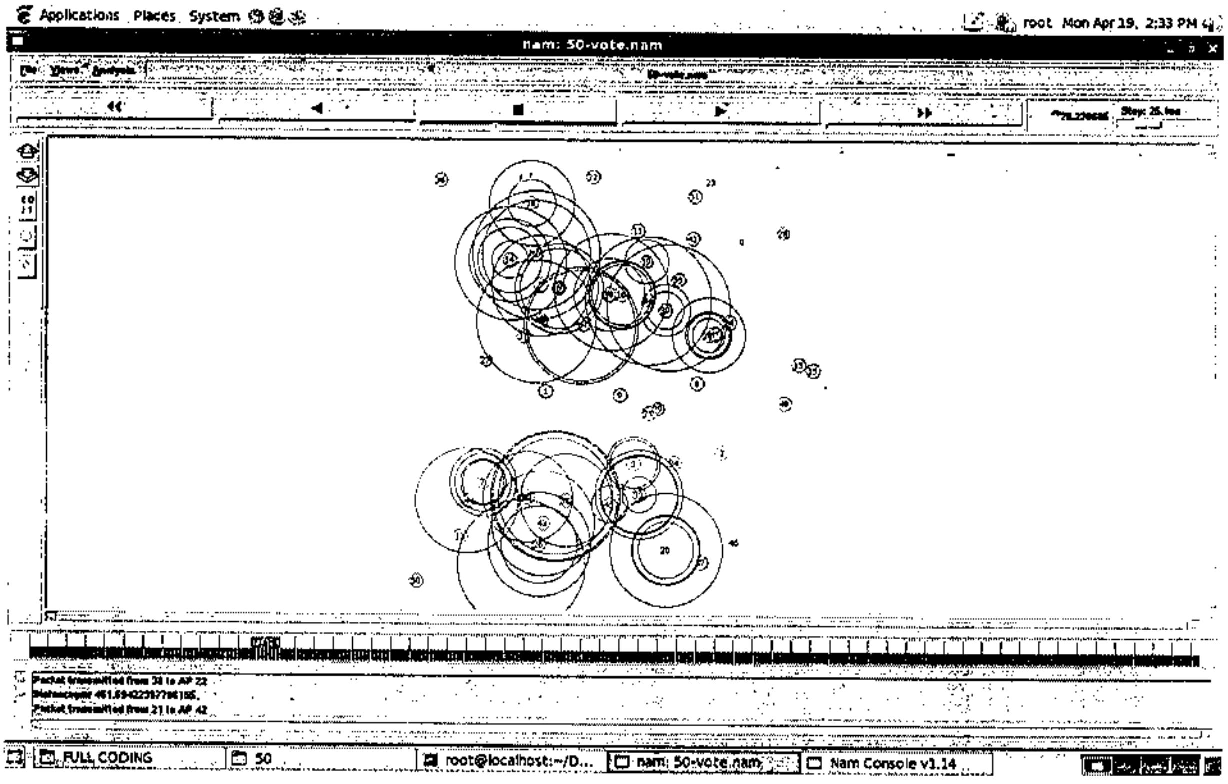
```
                                        set xc [expr $timea + 0.2]
                                        $ns_ at $xc "$ftp_(0) start"
                                        $ns_ at $xc "$cbr_($dest2(0)) start"
                                        $ns_ at $xc "$cbr_($dest2(0)) stop"
                                        $ns_ at $xc "$ns_ trace-annotate \"Packet
transmitted from $dest2(0) to AP $dest2($i) \""
                                        $ns_ at $xc "$ns_ trace-annotate \"
Distancegmr $dist_dd \""
                                   } else {
                                        puts "$dest2(0)---->$relay($dest2($i))"
                                        set tcp_(0) [new Agent/TCP/Newreno]
                                        $tcp_(0) set class_ 2
                                        set sink_(0) [new Agent/TCPSink]
                                        $ns_ attach-agent $node_($dest2(0))
$tcp_(0)
                                        $ns_ attach-agent
$node_($relay($dest2($i))) $sink_(0)
                                        $ns_ connect $tcp_(0) $sink_(0)
                                        set ftp_(0) [new Application/FTP]
                                        $ftp_(0) attach-agent $tcp_(0)
                                        set udp_($dest2(0)) [new Agent/UDP]
                                        $ns_ attach-agent $node_($dest2(0))
$udp_($dest2(0))
                                        set null_($dest2(0)) [new Agent/Null]
                                        $ns_ attach-agent
$node_($relay($dest2($i))) $null_($dest2(0))
                                        set cbr_($dest2(0)) [new
Application/Traffic/CBR]
                                        $cbr_($dest2(0)) set packetSize_ 512
                                        $cbr_($dest2(0)) set interval_ 4.0
                                        $cbr_($dest2(0)) set random_ 1
                                        $cbr_($dest2(0)) set maxpkts_ 10000
                                         $cbr_($dest2(0)) attach-agent
$udp_($dest2(0))
                                         $ns_ connect $udp_($dest2(0))
$null_($dest2(0))
                                         set xc [expr $timea + 0.2]
                                         $ns_ at $xc "$ftp_(0) start"
                                         $ns_ at $xc "$cbr_($dest2(0)) start"
                                         $ns_ at $xc "$cbr_($dest2(0)) stop"
                                         $ns_ at $xc "$ns_ trace-annotate \"Packet
transmitted from $dest2(0) to AP $relay($dest2($i)) \""
                                         puts "$relay($dest2($i))---->$dest2($i)"
                                         set dx [expr [$node_($relay($dest2($i))) set
X_]-[$node_($dest2($i)) set X_] ]
                                         set dy [expr [$node_($relay($dest2($i))) set
Y_]-[$node_($dest2($i)) set Y_] ]
                                         set dd [expr ($dx*$dx) + ($dy*$dy)]
                                         if {$dd<1} {
                                                 set dis_dd [expr (-1) * $dd]
                                                 set dist_dd [exprsqrt($dis_dd)]
                                         } else {
                                                 set dist_dd [exprsqrt($dd)]
                                         }
                                        set tcp_(0) [new Agent/TCP/Newreno]
                                        $tcp_(0) set class_ 2
                                        set sink_(0) [new Agent/TCPSink]
                                        $ns_ attach-agent
$node_($relay($dest2($i))) $tcp_(0)
                                        $ns_ attach-agent $node_($dest2($i))
$sink_(0)
                                        $ns_ connect $tcp_(0) $sink_(0)
                                        set ftp_(0) [new Application/FTP]
                                        $ftp_(0) attach-agent $tcp_(0)
                                        set udp_($relay($dest2($i))) [new
Agent/UDP]
                                        $ns_ attach-agent
$node_($relay($dest2($i))) $udp_($relay($dest2($i)))
                                        set null_($relay($dest2($i))) [new
Agent/Null]
                                        $ns_ attach-agent $node_($dest2($i))
$null_($relay($dest2($i)))
                                        set cbr_($relay($dest2($i))) [new
Application/Traffic/CBR]
                                        $cbr_($relay($dest2($i))) set packetSize_ 512
                                        $cbr_($relay($dest2($i))) set interval_ 4.0
                                        $cbr_($relay($dest2($i))) set random_ 1
                                        $cbr_($relay($dest2($i))) set maxpkts_ 10000
                                        $cbr_($relay($dest2($i))) attach-agent
$udp_($relay($dest2($i)))
                                        $ns_ connect $udp_($relay($dest2($i)))
```

```
$null_($relay($dest2($i)))
                                        set xc [expr $timea + 0.2]
                                        $ns_ at $xc "$ftp_(0) start"
                                        $ns_ at $xc "$cbr_($relay($dest2($i))) start"
                                        $ns_ at $xc "$cbr_($relay($dest2($i))) stop"
                                        $ns_ at $xc "$ns_ trace-annotate \"Packet
transmitted from $relay($dest2($i)) to AP $dest2($i) \""
                                        $ns_ at $xc "$ns_ trace-annotate \"
Distancegmr $dist_dd \""
                                   }
                              }
                         }
                         for {set i 0} {$i<$counter} {incr i} {
                              for {set j 0} {$j<$counter} {incr j} {
                                   set etree($i,$j) $otree($i,$j)
                              }
                         }
                    for {set i 0} {$i<$counter} {incr i} {
                              for {set j 0} {$j<9} {incr j} {
                                   set bit($i,$j) 0
                              }
                         }
                    for {set i 0} {$i<$counter} {incr i} {
                              for {set j 0} {$j<$counter} {incr j} {
                                   if {$otree($i,$j)==1} {
                                        set bit($i,$dest($i)) 1
                                        set bit($i,$dest($j)) 1
                                        set bit($j,$dest($j)) 1
                               }
                          }
                     }
set arrcnt 0
     set arrtop 0
     for {set i 0} {$i<$counter} {incr i} {
     set onecnt 0
          for {set j 0} {$j<9} {incr j} {
               if {$bit($i,$j)==1} {
               set onetmp [expr $onecnt + 1]
               set onecnt $onetmp
               }
          }
          if {$onecnt==1} {
               set end($arrcnt) $i
               set arrtmp [expr $arrcnt + 1]
          set arrcnt $arrtmp
          } else {
               set top($arrtop) $i
               set arrtmp1 [expr $arrtop + 1]
               set arrtop $arrtmp1
          }
 }
      for {set i 0} {$i<$counter} {incr i} {
 for {set j 0} {$j<$arrcnt} {incr j} {
               set etree($i,$end($j)) 0
```

```
}
}
set temp 99
for {set p 1} {$p<$temp} {incr p} {
            set selcnt 0
      for {set k 0} {$k<$arrtop} {incr k} {
      if {$top($k)==9999} {
} else {
                  for {set j 0} {$j<$counter} {incr j} {
                  if {$etree($top($k),$j)==0} {
                        set flag 1
                        } else {
                              set flag 0
                              break
                        }
                  }
                  if {$flag==1} {
                        set sel($selcnt) $top($k)
                        set seltmp [expr $selcnt + 1]
                        set selcnt $seltmp
                  }
            }
      }
      for {set j 0} {$j<$selcnt} {incr j} {
      for {set i 0} {$i<$counter} {incr i} {
                  if {$etree($i,$sel($j))==1} {
                        for {set d 0} {$d<9} {incr d} {
                              if {$bit($sel($j),$d)==1} {
                                              set bit($i,$d) 1
                                         }
                                    }
                         set etree($i,$sel($j)) 0
                              }
                         }
                         for {set q 0} {$q<$arrtop} {incr q} {
                              if { $top($q)==$sel($j) } {
                              set top($q) 9999
                              }
                    }
                    }
                    set flag2 1
          for {set i 0} {$i<$counter} {incr i} {
                         for {set j 0} {$j<$counter} {incr j} {
                              if {$etree($i,$j)==0} {
                                   ''set flag1 1
                              } else {
                                   set flag1 0
                              set flag2 0
                                   break
                              }
                         }
               if {$flag2==0} {
                              break
                         }
                    }
                          if {$flag1==1} {
                                break
                          }
                    }
                    for {set i 0} {$i<$counter} {incr i} {
                          if {$dest($i)==$first} {
                          }
              }
        }
              set u [expr $u + 3]
        }
  }
```

APPENDIX 2 – SCREEN SHOTS





```
Packet transmitted from 38 to AP 22
Distance... 461.894223177961185.
Packet transmitted from 21 to AP 42
```

REFERENCES

[1] Martin Mauve, HolgerF¨ußler, J¨orgWidmer, Thomas Lang, "Position-Based Multicast Routing for Mobile Ad-Hoc Networks," Institute for Mathematics and Computer Science – University of Mannheim, 2003.

[2]M. Transier, H. Füßler, J.Widmer, M.Mauve, andW.Effelsberg, "Scalable Position-Based Multicast for Mobile Ad-hoc Networks,"Technical Report TR-04-002, Department for Mathematics and Computer Science, Universityof Mannheim, 2004.

[3] J. Sanchez, P. Ruiz, X. Liu, and I. Stojmenovic, "GMR: Geographic Multicast routing for Wireless Sensor Networks," in *Proc. of IEEE SECON*, 2006.

[4] S. M. Das, H. Pucha, and Y. C. Hu, "Distributed hashing for scalable multicast in wireless ad hoc networks," *IEEE TPDS*, 2007.

[5] DimitriosKoutsonikolas, Saumitra Das, Y. Chârlie Hu, and Ivan Stojmenovic, "Hierarchical Geographic Multicast Routing for Wireless Sensor Networks,"Electronic, Electrical & Computer Engineering, The University of Birmingham, Purdue University, University of Ottawa, 2007

[6] Xiaojing Xiang andXin Wang, "A Scalable Geographic Service Provision Framework for Mobile Ad Hoc Networks" State University Of New York at Stony Brook, 2008