SOLONet: Sub-Optimal Location-Aided Overlay Network for MANETs*

Abhishek Patil[1], Yunhao Liu[2], Li Xiao[1], A.-H. Esfahanian[1], and Lionel M. Ni[2]

[1]*Dept of Computer Science and Engineering*
Michigan State University
East Lansing, MI 48824, U.S.A.
{patilabh,lxiao, esfahanian}@cse.msu.edu

[2]*Dept of Computer Science*
Hong Kong Univ of Science and Technology
Kowloon, Hong Kong
{liu,ni}@cs.ust.hk

Abstract

Overlay networks have made it easy to implement multicast functionality in wireless ad hoc networks. Their flexibility to adapt to different environments has helped in their steady growth. In MANETs, the position of nodes constantly changes; as a result, overlay multicast trees that are built using location information to account for node movement would certainly have a low latency. However, the performance gains of such a tree are offset by the overhead involved in maintaining precise location information. As the degree of (location) accuracy increases, the performance improves but the overhead required to store and broadcast this information also increases. In this paper, we present SOLONet, a design to build a sub-optimal location aided overlay multicast tree, where location updates of each member node are event based. Our simulation results indicate that such a sub-optimal tree does not compromise the performance gains of a location aided overlay multicast tree.

1. Introduction

Mobile ad hoc networks (MANETs) are characterized by mobile nodes and constantly changing network topology. Implementing multicast in such a dynamic environment is a challenging task. As pointed out by [18], traditional IP-layer multicast (e.g. [15], [19], [11]) for MANETs have a lot of signaling overhead as it needs to take into account the network dynamics in addition to the (multicast) group dynamics. The widespread deployment of IP multicast has been held back by a variety of issues [7]. Application layer (overlay) multicast relies on the underlying unicast protocols to adapt to the changing network topology. As a result, the application layer has to track only the group dynamic. Due to its ease of implementation and flexibility to adapt, overlay multicast networks (though not as efficient as IP layer multicast) are finding many practical applications in MANETs. AMRoute [4], PAST-DM [9], and LGT [5] are some of the overlay multicast protocols that have been proposed for MANETs. In recent years, we have seen a dramatic increase in research interest shown towards context aware computing and location-sensing techniques [2], [10], [14], [26], [16], [20]. Consequently, position based approach for routing (e.g. [5], [21]) is becoming practical. This paper presents SOLONet, a design that uses location information of member nodes to build an overlay multicast tree.

Application layer multicast is not as efficient as IP-based multicast. As can be seen in Figure 1(b,c), data exchange between member nodes requires traversing other member nodes. The latency increases as the number of nodes increases. This delay can be greatly reduced when the overlay tree is built by taking into account the member node positions (Figure 1c). Such a tree would keep track of member node's movement and would be frequently updated to account for any change in the node positions. The nodes that are physically close to each other would be neighbors in the logical tree (Figure 1c) and the logical distance of any member node from the source node will be proportional to its actual distance from the source.

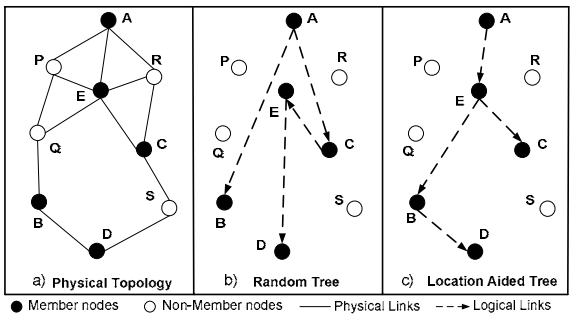

Figure 1: Random vs location-aided overlay tree.

* This work was partially supported by the US National Science Foundation (NSF) under grant ACI-0325760, by Michigan State University IRGP Grant 41114 and by Hong Kong RGC Grants HKUST6161/03E, HKUST6264/04E, and DAG02/03.EG02.

It is quite clear that a location aware overlay tree would give considerable improvements in the performance over any random overlay tree (Figure 1b). However, there are several issues with this approach that need close attention. How to effectively distribute the location information of each member node to other members? How precise should the location information be? How often should this information be updated? Location broadcast is a costly affair and if every node starts to broadcast its current location information, then it would quickly lead to the broadcast storm problem [17] [22]. Several papers have proposed methods to carry out broadcast with minimum overhead. Wu and his team [23] [24] have suggested a number of optimizations (using the concept of dominating set) to carry-out efficient broadcast in MANETs. The method proposed in our paper builds location-aided overlay trees without the need for any member node to broadcast its location information.

Intuitively, precise location information would lead to a better overlay tree. However, obtaining exact location of an object is a difficult task, especially in indoor environments where ad hoc networks are usually implemented. Frequent updates would be necessary to maintain the accuracy of this information. As the node number and mobility increases, the update frequency would exponentially increase. There is a tradeoff between the advantages of building a location-aided overlay tree versus the distribution and precision of this location information. In SOLONet, the physical topology is divided into smaller areas (cells) having a certain geometric shape (e.g., triangle, square, hexagon, etc.). The idea is to make location updates event-based. A node will report a change in its location only when it crosses border to a neighboring cell. A leader node is selected in each cell to localize certain operation, aid in service discovery and to reduce the number of broadcast messages. A node wishing to form or join an existing overlay network (for a particular service) would query its local leader instead of broadcasting the query to each node in the network. The leader node also maintains information about each node in its cell. The amount of information maintained depends on a particular implementation. Using NS2 (for simulations), we compare the performance of our design with an optimal overlay tree. We also evaluate the scalability and performance for different member size and cell areas. We also look at the effects of different location update frequencies on performance.

The rest of the paper is structured as follows. Section 2 summarizes previous work on overlay multicast. Section 3 presents an in-depth description of our design. Section 4 presents a detailed analysis of our leader selection algorithm. In Section 5 we present our simulation results. Finally, Section 6 concludes the paper and presents directions for future research.

2. Related Research.

Several overlay multicast protocols (e.g. [9] [5] [6] [3] [1]) have been proposed and studied in recent past. Many of them have addressed the issue of building an efficient overlay multicast tree. The NICE [3] project aims to address the issues involved in data stream applications – real-time data applications that are characterized by a very large set of receivers having low bandwidth. NICE arranges the end host into sequentially numbered layers, which defines the multicast overlay data path. The basic operation of NICE is to create and maintain a hierarchy consisting of a set of end hosts. The members at the top of the hierarchy maintain state about O(log N) other members, where N is the number of nodes in the network. Member nodes keep information about members 'near' to them in the hierarchy and have limited knowledge about other members. This structure helps localize the effects of a member failure. Hosts at each layer are partitioned into clusters that have a cluster leader. The leader selection in NICE does not make use of any location or the battery strength information. In NICE, each cluster size depends on the set of hosts that are close to each other; whereas, in our approach, the cell edges (physical boundaries) define the membership. NICE is not defined for MANETs and hence it does not take into account node movement.

Progressively Adaptive Sub-Tree in Dynamic Mesh PAST-DM [9] is an overlay multicast protocol defined for MANETs. It tries to eliminate redundant physical links so that the overall bandwidth consumption of the multicast session is reduced. The virtual mesh in PAST-DM constantly adapts to the changes in the underlying network topology. Each node implements a neighbor discovery protocol using the extended ring search algorithm. The nodes periodically exchange link state information with their neighbors in a non-flooding manner. Thus, by looking at the link state of each node, a node gets a view of the entire topology. This information is used to build a source-based tree. PAST-DM yields a stable tree quality at the cost of higher overhead, which increases with the periodicity of the link state updates.

Location Guided Tree (LGT) [5] is a small group multicast scheme similar to DDM [12]. It builds overlay multicast trees (in MANET) using geometric distance as the heuristic of link costs. The scheme proposes two tree construction algorithms: greedy k-ary tree construction (LGK) and Steiner tree construction (LGS). The algorithms are based on the assumption that longer geometric distances require more network-

level hops to reach destination. LGS constructs the Steiner tree using link costs as their geometric lengths. With LGK, source node selects k nearest neighbors as its children and partitions the remaining nodes according to their distance to the children nodes. Similar to LGT, our approach uses location information to form an overlay tree. However, in our approach, the location updates are event based and hence not very frequent. LGT is a small group multicast scheme and may not scale well.

3. Design of SOLONet

Before getting into the details of the design, this section will discuss various components involved.

3.1. Location and Geometry Identification

We assumes that each node knows the topology (and the cell structure) of the given area. Organizers of special events (like football game or a concert), where overlay multicast may be implemented, may distribute a coordinate file or a hash function that defines the topology. The users of the mobile device, who wish to participate in the multicast network, may download the file from the organizer's website or request a copy from a neighboring node that has already joined the network. We also assume that each node knows its current location. This information may be relative to some point in the topology and can be computed using the hash function or the coordinate file. To monitor their movement, nodes may use location sensing techniques like GPS in outdoor environment or one of the several indoor location sensing techniques (e.g. [2], [10], [14], [26], [16], [20]) available.

3.2. Cell Identification

When the ad-hoc network comes online for the first time, it would run some form of an IP allocation algorithm (e.g., Prophet Algorithm [27]) to get a unique IP address. This IP address will be used to identify each node. In the next few sections, we discuss how the topology is partitioned into disjoint cells.

3.2.1. Shape of the cell. Our design doesn't put any restriction on the shape of the cells. In theory, the topology can be divided into several (non-overlapping) cells of any shape. However, by using regular cell structures (which repeat over the entire area), we can make use of the geometric properties of that shape. Each shape will have its own advantages. A hexagon can closely resemble a coverage area for a given cell, while a square shape is easier to divide into smaller areas if the density of nodes in a cell is high.

3.2.2. Size of the cell. Our design requires that all nodes in any cell are in the coverage area of each other. For example, in a square, two the ends of a diagonal are points that are farthest apart. Therefore, if the square cells were chosen, then the size of the cell should be such that the length of the diagonal is less than the coverage radius. This will ensure that all the nodes in the cell are within each others coverage. Similar calculations can be done for cells of other shapes using the geometric properties of their shape.

3.2.3. Center of the cell. Since the cell topology would be known to each node, computing the center of its current cell would not be difficult (using some geometric property of the cell's shape). During this phase, the nodes perform some geometric calculation (depending on the method used for representing their location) to determine which cell it is current present. This information will help them set their cell ID (CID) parameter. The CID would change when the node moves to a different cell.

3.3. Member Nodes and Leaders

Each cell would have a local leader to assist the tree building process. This section gives an overview of the responsibilities of a member and a leader node.

3.3.1. Node responsibilities. When a node enters a new cell, it defers it's *disconnect* message to the old leader till it is able to connect to the leader in the new cell. The node waits for a time equal to the periodicity of the beacon (Section 3.3.2) in an attempt to know who the local leader is (Figure 2a). After receiving the beacon message, this node would know the leader's IP address or other relevant details (Figure 2b). The node would then send an association message to the new leader. The scenario where the leader's beacon message is lost or the neighboring cell has no leader is discussed in later sections.

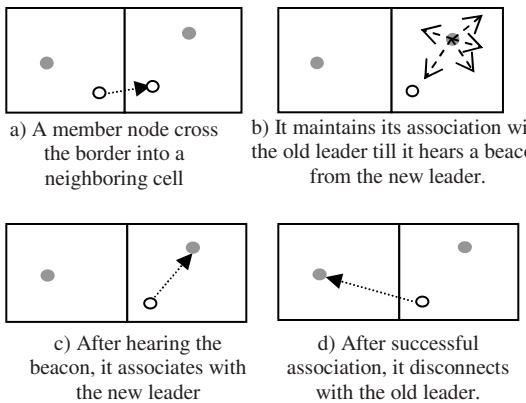

Figure 2: Node association with a new leader after crossing cell boundary.

Table 1: Typical state table at each leader node

Node ID	X	Y	Type of Service	Battery strength	Time in Cell
12	76.22	108.37	Gateway to Internet	50%	4 min 27 sec
57	73.76	111.29	Live Surgery Video	89%	1 min 12 sec
23	75.32	105.12	Medical Record Files	24%	4 Sec

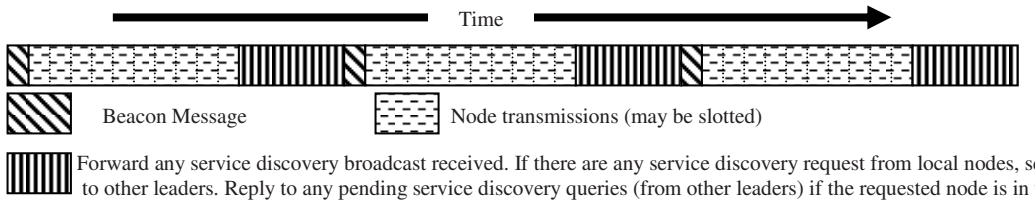

Figure 3: Time-line showing all the communication happening in a cell.

After successfully associating with a new leader, the node disconnects from the old leader (Figure 2c, 2d). This deferred disconnect procedure ensures that a node is always connected to at least one leader. Since the node has just crossed the cell boundary, it is most likely going to be in the coverage of the old leader. After association with the new leader, the node provides it with information about its current location, battery power and service that it can provide. After the first update, all subsequent updates to this leader carry only the battery and current location information.

3.3.2. Leader responsibilities. As explained in the previous section, nodes constantly update the local leader with information about their location (and service type, battery strength, etc). A leader node would maintain a table containing information about each node in its cell. The leader node maintains this information till it receives a *disconnect* message from a node that has left its cell. Rows in such a table may look like the ones shown in Table 1. The leader node periodically broadcasts a *beacon* packet to all the nodes in its cell. This beacon message aids in leader selection process, serves as a feedback to the nodes and indicates that the leader node is alive. The beacon packet is discussed in detail in Section 4.2. Figure 3 shows a typical communication time-line in every cell. Storing service type of each node is important for service discovery by other nodes in the network.

Additional responsibilities of a leader could be time slicing of node transmissions or cell splitting. If the node density in a cell is higher than a certain threshold, the leader may assign time slots to each node to avoid collisions between their transmissions. These time slots may be carried in the beacon message sent by the leader (Figure 3). Another alternative to solve the high cell density problem could be cell splitting. We do not discuss cell splitting in this paper due to page limitation. The next section gives a detailed description of the role a leader node plays in service discovery and how a location-aided overlay tree is formed.

3.4. Service Discovery

When a member node wishes to get a particular service, it would query its local leader. The node may provide the address (if known) of the source node that provides the requested service. For example, there may be a few nodes in the network that act as gateway nodes and provide access to the Internet. A node that wishes to access the Internet may request its local leader to find a gateway node. If the requesting node knows the IP of a gateway node, it may provide the IP along with its request to the local leader nodes. After receiving such a request, the leader node checks its local table (similar to Table 1) to see if the source node is in its list (i.e., in the same cell). If the requested node is not found locally, the leader broadcasts (forwards) the request to all other leaders (Figure 4b). It must be noted that this message exchange between leaders may be multi-hop communication as the leader nodes may not have a direct link between them. In this broadcast message, the leader node provides IP address of the requesting node and the CID. We have tried to keep the broadcast message as small as possible by having only two items (IP and CID) in the message. This broadcast is only between leader nodes. Since there are very few leader nodes in the entire network the broadcast overhead will be very low. Each leader node, upon receiving such a broadcast, checks their local node list to see if the requested (service) node is in its cell. If the requested node is found, then the leader forwards the request (message) to that node (Figure 4).

In the event that the requesting node doesn't get any response for a timeout period, it resends its request. Certain request may not generate any reply either because there is no node in the entire network that can provide the requested service or because the

leader in the requested node's cell is dead (Figure 7). Both of these problems can be solved if the timeout period follows an exponential back-off scheme. Such a scheme would give enough time for the leader selection process to complete in requested cell or provides enough time for a new node (that provides the requested service) to settle down (i.e. associate with a local leader and register its service, etc). The system can be configured so that after a certain number of timeouts, a node stops making request for that service.

3.5. Joining an Overlay Tree

The node that provides services (like data storage, access to Internet, computing resources etc) to member nodes in the multicast tree is referred to as the source node. The source node is the root of the multicast tree built for providing a particular service. It stores the CIDs of the member nodes that it is currently serving. When a source node receives a request for service (from a new member), it extracts the CID (from the request message) and finds the position of the requesting node. The position is assumed to be in the center of the requestor's cell. Simulations in Section 5.1 investigate the performance with this assumption. The source node now checks its internal tree-table to find node(s) that may be in the same cell or neighboring (closest) cell as the requesting node. It now contacts the requesting node and provides it with the IP address of a member node nearest to it. The source also provides appropriate information to this 'nearest' node about the requesting node (Figure 4c). With this information, the requesting node can now connect to a near-by (physically close) member node, which would in turn provide the required service (Figure 4d). As our simulations confirm, the latency with this approach is much lower compared to the case where a source node randomly selects a node in the tree for the requestor to connect to.

In case of prioritized overlay multicast [25], the source node will also provide information about the priority of the service that it provides. This would help in the formation of a prioritized overlay tree. At the time of request, the requesting node may be part of some other group(s) having a different priority. For example, in a particular hospital, all doctors, nurses or resident students doctors may have their own category (viz *doctor_net*, *nurse_grp*, *student_org*) in addition to a common (low priority) group called *hospital_net*. In an emergency, a group of doctors, nurses and residents may form a high priority network to address a specific patient case. Priority tree and its formation is discussed more in detail in [25].

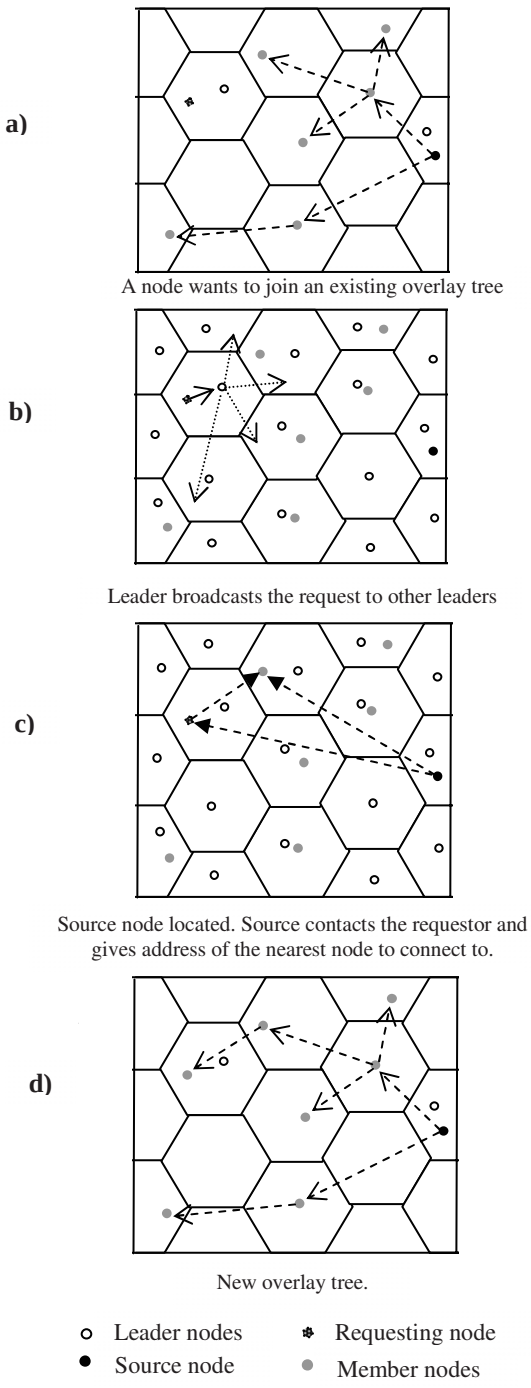

Figure 4: Overlay tree with location information.

3.6. Event-based Update

As mentioned earlier, every source maintains a list of member nodes that it serves (directly or through other members). A member node updates the source with its new location (CID) only if it changes its cell. This is an important contribution of our design – an

event-based update; the event being crossing the cell boundary. This approach reduces the amount of location updates and the associated overhead. The source node can alter the tree structure depending on the updates that it receives. Although the resulting (location aware) tree is sub-optimal, one can appreciate the gains of this method if it were to be compared with the expensive broadcast approach needed when precise location information in desired. Simulation results in Section 5.1 show that there is not much of a performance penalty when the source assumes the center of a cell (as its location for each node) for building (or reorganizing) the overlay tree.

4. Leader Selection for Cells

Before we get into the details of the leader section process, Sections 4.1 and 4.2 look at some of the arrangements in our design that aid this process.

4.1. Responsibilities of Leaders

Every leader performs activities that can aid in the selection of a new leader in case of its failure. In addition to the location and the service information, each leader also maintains battery status and the time a particular node spent in its cell (Table 1). Weights are assigned to the battery strength, time-in-cell and node's current location information. The entries in the list are sorted according to the result of this weighting function. This sorted list is broadcasted in the beacon message (Section 4.2). The first two nodes or the top 10% nodes (whichever is greater) in this sorted list are called *candidate nodes* – meaning that these nodes are potential candidates for leadership in case the current leader fails. Listed below are some important parameters used to choose the candidate nodes.

4.1.1. Time information. It has been observed in [8] that hosts that have been stationary for a period of time are more likely to remain stationary as compared to those currently in motion. Thus choosing a node that has shown little movement or no movement as the leader would greatly increase the chances that it would stay in that cell for a long time.

4.1.2. Battery strength. A leader has a lot of responsibilities and this demands battery power. A node which has good battery strength should be selected as the leader, so that it can perform the leadership responsibilities without interruption for a long time.

4.1.3 Location. A leader situated more or less towards the center of the cell can serve all the cell nodes with little delay. Even if this node were to be in motion, it would take a longer time for it to move out of the cell due to its distance from the cell's periphery.

The next section shows how the sorted list is made available to all the nodes in the cell through the beacon packet (Figure 5).

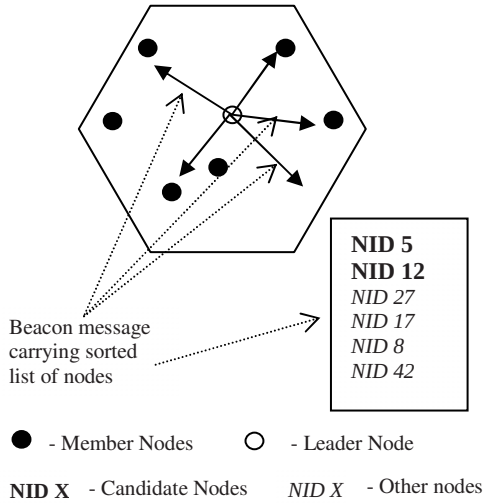

Figure 5: Beacon message from the leader node.

4.2. Beacon Message

Every leader periodically broadcasts a beacon packet containing a sorted list (described in 4.1) of nodes in its cell. These beacons serve three purposes.

4.2.1. Leader's heartbeat. Beacon is a way for the leader to tell the other nodes that it is alive. If nodes do not receive beacons for a pre-configured timeout period, they would suspect that the leader is no longer available. The use of timeout will help prevent false detection. A beacon packet may be lost due to noise, multi-path fading, or collision with some other transmission. Noise and fading depend on environmental conditions while collision (although rare) may depend on density of member nodes. Collision can be reduced by assigning time-slots to each node (Section 3.3.2).

The timeout value is not fixed and will depend on several factors (e.g., noise and collision rate in that environment). It will be available to the nodes at start up when they acquire the topology coordinates (or hash function). The leader node may become unavailable for the following two reasons. The user 'pulled the plug' – turned off the device in an unconventional manner (e.g., suddenly removed the batteries). In such a scenario, the leader node would die without informing any other node. The other case is when a leader sends a message saying that it is stepping down but the message was lost (perhaps due to noise or collision). The use of periodic beacon will help detect the above two scenarios.

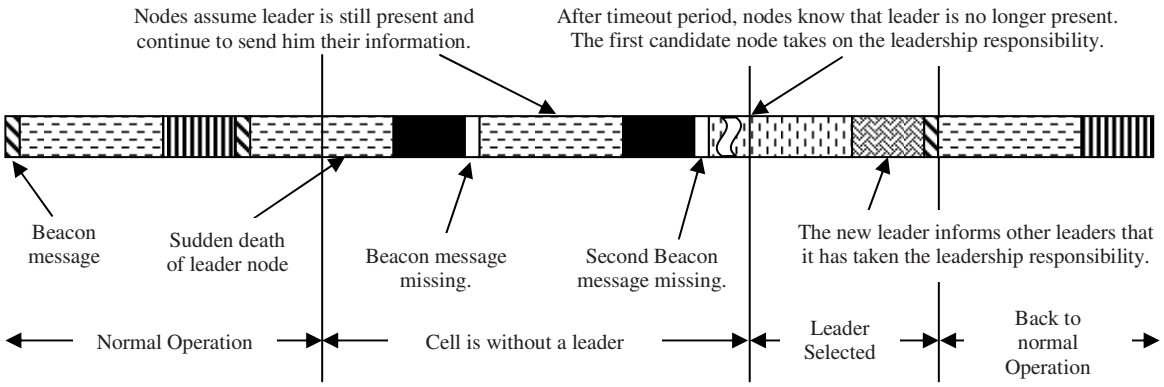

Figure 6: Communication time-line – failure of a leader and selection of a new leader.

4.2.2. Aid in leader selection. The beacon message contains the sorted list of nodes (Figure 5). Leader failures are very rare; however, in case the current leader fails, all the nodes examine the sorted (node) list that came in the previous beacon broadcast. The first two nodes in the list are the candidate nodes i.e., nodes that can become leaders. The first candidate node has to acknowledge that it is taking leadership before the next beacon period. If it fails to do so, the second candidate node sends a broadcast declaring itself as the leader. The possibility that both the nodes become unavailable at the same time is very rare and if it happens, then after another timeout, the next candidate node in the sorted list takes over the leadership.

4.2.3. Feedback to each node. The beacon message can serve as a feedback to each node to indicate that its message (containing location and battery information) reached the leader without any error. The beacon message would help identify the IP address of the new leader node when a node crosses over into a new cell. In case of a crowded cell, the beacon may also contain the time slots telling each node when to transmit its information thus avoid collisions between two (or more) nodes. Implementing slotted transmission in a non-dense cell is optional. Nodes can also use the beacon to synchronize their clock with the leader node.

4.3. Hello!! Anybody Home?

This section describes a worst case scenario. When a node crosses border and enters a neighboring cell, it maintains its association with the old leader and waits for the beacon from the new leader. What if there is no leader in the new cell or if the leader there died? In our design, the node waits a little longer than the timeout period mentioned above. This would give other nodes (originally present in that cell) enough time to take over the leadership. After this long wait, if there is still no sign of a beacon from a leader, the node assumes that the neighboring cell was empty – no leader nor member nodes were present. The node now broadcasts a message containing its IP and its desire to become the leader (Figure 7a). Since the cell size is such that any node in that cell will hear this broadcast, the node waits for a small timeout period for response from any other node that might recently enter the cell (Figure 7b). If there was another node that entered the cell at approximated the same time, the node with the lower IP would take over the leadership. After the node has taken the leadership responsibility, it sends a *disconnect* message to the old leader. During this leader election process, the node was still associated with the old leader. Since the node has recently crossed the border, the inaccuracy in its location information would be similar to the location inaccuracy for a node in the older cell which is near the border of the cell.

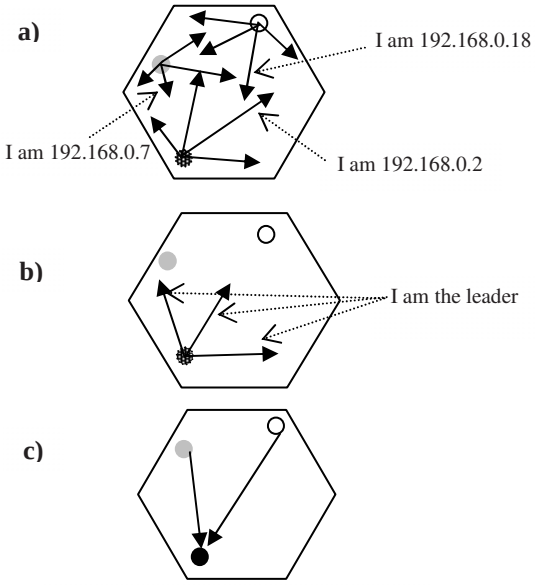

Figure 7: Leader selection during initialization or when no leader is present in a new cell.

4.4. Initiation of Leader Selection

There are three scenarios when a leader selection is required as described below:

4.4.1. When the network first comes online. When the network first comes online, the nodes follow the procedure mentioned in Section 4.3 to decide who would be their local leader. The first leader may not be the best leader in terms of battery power, location and other factors. However, one of the responsibilities of the leader node is to find a good replacement leader, hence subsequent leaders would be wisely chosen.

4.4.2. Leader wishes to give up leadership. There can be different cases that may cause a leader to give up its leadership – proximity to the cell boundary, running low on battery or the user has gracefully switched off (shut down) the mobile device. After it has decided to quit, the leader sends a broadcast message informing all the nodes. One of the *candidate nodes* takes over the leader responsibilities.

4.4.3. Exceptional cases – rare in occurrence. Leader's quit message was lost due to collision or noise or the user switch off the device in an unconventional manner. Both these conditions are seldom possible and would be detected by the absence of beacon messages for a timeout period. The leader selection strategy is the same as that described in Section 4.2.2.

5. Simulations

Simulations were carried out using NS2.26. As of this writing, NS2 doesn't have any extension for simulating overlay multicast in MANETs. With the help of C-programming and bash scripting, the traffic pattern generated by CMU's cbrgen utility was modified to represent a location-aided overlay network. The setdest utility was used to generate different node positions and movement patterns. The nodes in the simulation move according to the 'random waypoint' model [13]. The first set of simulations compares an optimal tree (which is built by using precise location information of member nodes) and our proposed sub-optimal (SOLONet) overlay tree. This comparison is done for two different areas: 500x500m^2 and 800x800m^2. The second simulation set aims to show the scalability of our design. We compare the performance for 10, 15, 20 and 30 member nodes. The third simulation set compares the performance for difference choices of cell area and for different number of member nodes. We compare the performance for 25x25, 50x50, 100x100 125x125 and 250x250m^2 for a topology of 500x500 m^2. In all the three scenarios, the file size used for transfer is 50KB and the packet size is 512 bytes.

5.1. Optimal vs Sub-Optimal (vs Random)

Each node updates its local leader with information about their current location. The periodicity of this update determines the accuracy of the location information present at the local leader. This location information will be used during the formation of the location-aware tree. As mentioned earlier, there is a trade off between the frequency of updates and the overhead involved. With lower update times, the node information will be most current at the leader nodes. However if each node were to initiate frequent updates, the network would be swamped with update packets. This section presents results of simulations for different update times. The simulation is performed for an area of 500x500m^2 and 800x800m^2. The cell size in both cases was chosen to be 100x100m^2. The movement pattern was 5 m/sec with a pause time of 10 sec. The total number of nodes was set to 150 and the number of member nodes was kept at 15. These nodes used center of their (respective) cells as their location during the formation or the joining of the overlay tree. In each case, the (tree building) start time was a randomly chosen value between 0 and the update value (chosen for that particular simulation scenario). For example if the update value chosen was 70 sec, then the start time would be randomly distributed between 0 – 70 sec.

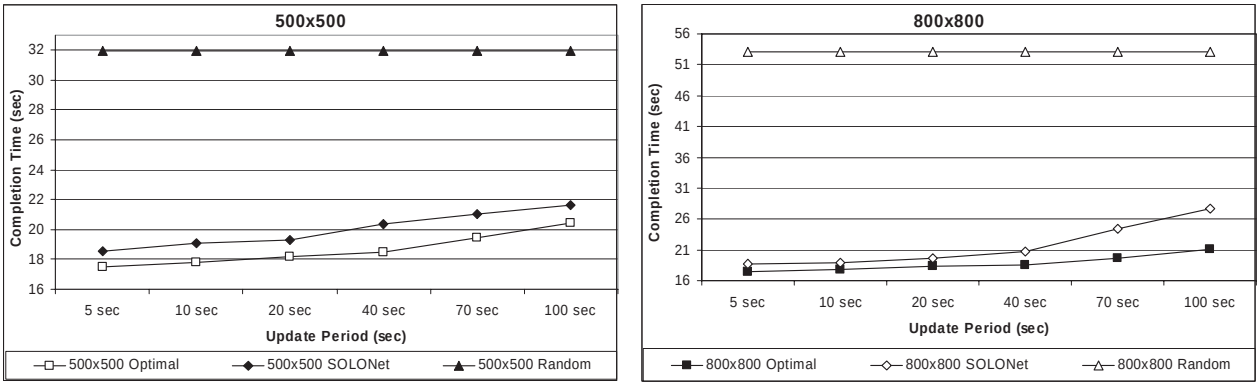

Figure 8: Comparison between accurate location information and location reported as the center of the cell.

By having start-up time between 0 and update time, we try to simulate the situation where an overlay tree was built using location information that was updated "*start-time*" sec before. Each simulation result is the average of 50 different scenarios. From Figure 8, it is clear that the performance of a sub-optimal tree closely matches with that of an optimal tree. Figure 8 also shows the performance when no location information is used (i.e., a random overlay tree).

5.2. Scalability Consideration

We repeated the above simulations for an area of 500x500m^2 for different number of member nodes – 10, 15, 20, and 30 nodes. The update time of 20 sec was chosen – which meant that the nodes updated their location information randomly in 0 to 20 sec. The results in Figure 9 show that the completion time increases with the increase in the member nodes. This was expected. The simulation also ascertains the scalability of our design – the performance of optimal and a sub-optimal tree is very close.

5.3. Effects of Smaller Cell Size

The aim of this set of simulations was to find a relation between the performance and cell size. Cell sizes of 25x25, 50x50, 100x100 125x125 and 250x250m^2 were checked. The topology area was chosen to be 500x500m^2. The simulations also had varying member size – 10, 15, 20 and 30 members. The movement pattern was 1 m/sec with a pause time of 10sec. It is easy to see from the result in Figure 10 that the performance holds an inverse relation with the cell size. Smaller area gives an improvement in the performance. This is because smaller areas imply higher accuracy in the location information used for building the location tree.

However, it should be noted that this improvement in performance is offset by an increase in the service broadcast overhead. When the topology is divided into large number of smaller cells, any broadcast (during service discovery – Section 3.4) would go through more leaders and will take longer to reach the entire leader set. This would generate a very high overhead. As a result, the service discovery process will be slower and very inefficient. On the other hand, with larger cells, the location accuracy is lowered but the communication overhead and the propagation delays are reduced. Thus, there is a tradeoff between the overhead in service discovery and the performance of the overlay tree. The discussion about optimizing this tradeoff is beyond the scope of this paper and is part of our future work. Smaller cell size is not recommended if the node mobility is high. This is because, with high mobility, nodes will constantly cross cell boundaries triggering frequent location updates.

There was a significant degradation in the performance when the entire topology (500x500m^2) was viewed as one big cell. The values noted are shown in Table 2. Since all the nodes (including the source node) report the center of the cell as their location, the source node finds all of the nodes as nearest to itself - resulting in a star topology, where the source node makes direct connections with all the requesting nodes. This increases the overhead (and collisions) at the source node resulting in a considerable loss of performance.

Table 2: Completion time values for cell size of 500x500 (entire topology)

Nodes	10	15	20	30
Time	29.5	67.5	140.4	256.8

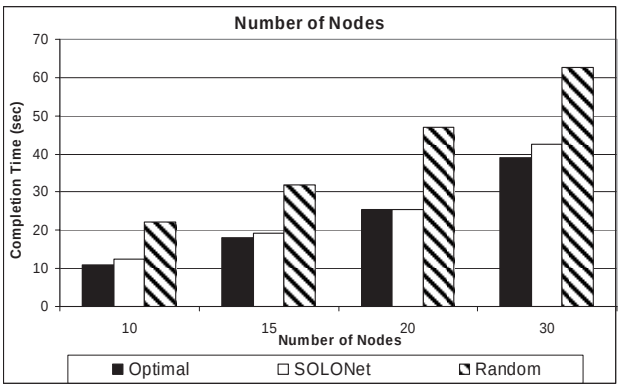

Figure 9: Scalability of the protocol.

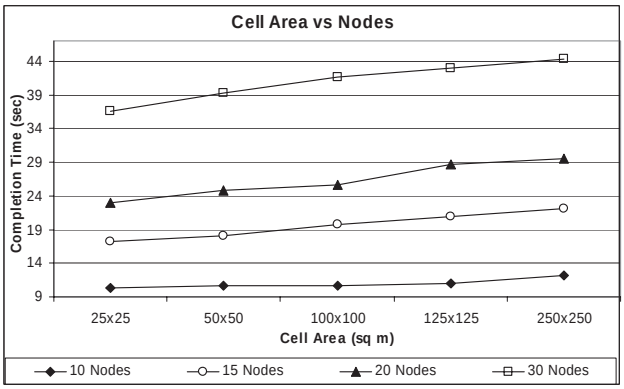

Figure 10: Performance of SOLONET for different cell sizes and member nodes.

6. Conclusion and Future Work

This paper presents SOLONet, a design to build sub-optimal overlay multicast trees, where the physical topology is divided into smaller cells having a local leader to perform various tasks. The paper also gives a detailed description of local (cell) leader selection process and the use of a beacon message for carrying out various activities in a cell (including leader selection). Our simulation results show that SOLONet is scalable and its performance closely matches that of an optimal overlay multicast tree. It was also observed that smaller cell area gives better performance as the location accuracy increases. In our current simulations, we have considered square cells; however, in the future, we plan to test SOLONet with other shapes.

As a future direction, we are exploring ways to construct a degree bounded location aware overlay multicast tree which would have minimum single-point-failure nodes. We are also investigating an algorithm that can adaptively divide a cell into smaller sub-cells if the density of nodes increases beyond a certain value. Smaller cell size gives better performance but higher broadcast overhead. We are looking at ways to reduce the service discovery broadcast overhead small size cells.

7. References

[1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "The case for resilient overlay networks," 8th Annual Workshop on Hot Topics in Operating Systems (HotOS-VIII), May, 2001.

[2] P. Bahl and V. N. Padmanabhan, "RADAR: An In-building RF-based User Location and Tracking System," IEEE INFOCOM, March, 2000.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," ACM SIGCOMM, August, 2002.

[4] E. Bommaiah, M. Liu, A. MvAuley, and R. Talpade, "AMRoute: Ad Hoc Multicast Routing Protocol." Internet Draft, draft-manet-amroute-00.txt, March, 2002.

[5] K. Chen and K. Nahrstedt, "Effective Location - Guided Tree Construction Algorithm for Small Group Multicast in MANET," IEEE INFOCOM, May, 2002.

[6] Y. Chu, S. Rao, and H. Zhang, "A Case of End System Multicast," ACM Sigmetrics, June, 2000.

[7] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," in IEEE Network Magazine, Jan-Feb, 2000.

[8] R. Dube, C. D. Rais, K. Wang, and S. K. Tripathi, "Signal stability based adaptive routing (SSA) for ad hoc mobile networks," IEEE Personal Communication, February, 1997.

[9] C. Gui and P. Mohapatra, "Efficient Overlay Multicast for Mobile Ad Hoc Networks," Wireless Communications and Networking Conference (WCNC), March, 2003.

[10] J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian Filtering for Location Estimation," IEEE Pervasive Computing, July-Sept, 2003.

[11] J. Jetcheva and D. B. Johnson, "Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks," MobiHoc, October, 2001.

[12] L. Ji and M. S. Corson, "Differential Destination Multicast - A MANET Multicast Routing Protocol for Small Groups," IEEE INFOCOM, April, 2001.

[13] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," Mobile Computing, 1996.

[14] H. Koshima and J. Hoshen, "Personal Locator Services Emerge," in IEEE Spectrum, February, 2000.

[15] S. J. Lee, M. Gerla, and C. C. Chiang, "On Demand Multicast Routing Protocol," IEEE WCNC, September, 1999.

[16] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," IEEE PerCom, March, 2003.

[17] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu, "The Broadcast storm problem in a mobile ad hoc network," 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), August, 1999.

[18] K. Obraczka, G. Tsudik, and K. Viswanath, "Pushing the Limits of Multicast in Ad Hoc Networks," IEEE ICDCS, April, 2001.

[19] E. M. Royer and C. E. Perkings, "Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol," ACM MOBICOM, August, 1999.

[20] A. Smailagic, D. P. Siewiorek, J. Anhalt, D. Kogan, and Y. Wang, "Location Sensing and Privacy in a Context Aware Computing Environment," in Pervasive Computing, 2001.

[21] I. Stojmenovic, "Position based routing in ad hoc networks," in IEEE Commmunications Magazine, vol. 40, July, 2002,, pp. 128-134.

[22] Y. C. Tseng, S. Y. Ni, and E. Y. Shih, "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks," IEEE 21st International Conference on Distributed Computing Systems, 2001.

[23] J. Wu, "Dominating-Set-Based Routing in Ad Hoc Wireless Networks," in Handbook of Wireless Networks and Mobile Computing, I. Stojmenovic, Ed.: John Wiley & Sons, 2002, pp. 425-450.

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

[25] L. Xiao, A. Patil, Y. Liu, L. M. Ni, and A.-H. Esfahanian, "Prioritized Overlay Multicast in Ad-hoc Environments," IEEE Computer Magazine, Feburary, 2004.

[26] M. Youssef, A. Agrawala, and A. U. Shankar, "WLAN Location Determination via Clustering and Probability Distributions," IEEE PerCom, March, 2003.

[27] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet Address Allocation for Large Scale MANETs," Ad Hoc Networks Journal, November, 2003.