LOCATION MANAGEMENT IN PEER-TO-PEER MOBILE ADHOC NETWORKS

Mujtaba Khambatti and Sarath Akkineni

Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85287-5406

Technical Report
May 2002

ABSTRACT

The combination of peer-to-peer systems and ad-hoc networks allows for the creation of highly dynamic, self-organizing, mobile P2P systems. By taking advantage of location information, a routing protocol in these systems can aid the mobile hosts to efficiently communicate with each other. There is a need therefore for highly scalable location management systems that can provide location information of any mobile host at any time. However, location management is a hard problem. The goal of a good location management scheme should be to provide efficient searches and updates. Updates occur when a mobile host changes location; searches occur when a mobile host wants to communicate with a mobile host whose location is unknown to the requesting host.

In this report, we describe two novel location management strategies: hierarchical and de-centralized. The hierarchical strategy assumes the existence of a small number beacons that act as both base stations and location servers. Beacons are logically connected in a binary tree topology and are mobile like the mobile hosts. The de-centralized approach employs a version of a well-known peer-to-peer search technique to achieve location management amongst mobile hosts without the existence of beacons. We evaluate the two strategies though simulations and compare them based on their effectiveness in enabling a greater number of successful transmissions between the mobile hosts.

1. INTRODUCTION AND OVERVIEW

Mobile computing has emerged as one of the leading technologies in the 21st century and it is still rapidly growing. It provides users with the capability of accessing information regardless of their location. Another technology that has been getting a lot of attention lately is peer-to-peer systems. Peer-to-peer (P2P) systems are distributed systems, in which nodes of equal roles and capabilities exchange information and services directly with each other [4, 1]. P2P architectures differ from client-server architectures, in which some computers are dedicated to serving the others.

The combination of peer-to-peer systems and ad-hoc networks allows for the creation of highly dynamic, self-organizing, mobile P2P systems. Mobile hosts continuously change their physical location and establish peer relationships among each other. In order to provide end-to-end communication throughout the network, mobile hosts must cooperate to handle network functions. The absence of any centralized, dedicated servers to maintain the location information of the mobile hosts in P2P ad-hoc networks poses a challenge. Therefore location management becomes an important issue.

Location Management consists of location updates, searches and search-updates [11]. Updates occur when a mobile host changes location; searches occur when a mobile host wants to communicate with a mobile host whose location is unknown to the requesting host; and search-updates occur after a successful search, when the requesting host updates the location information corresponding to the mobile host. The goal of a good location management scheme should be to provide efficient searches and updates. The number of messages sent, size of messages and the distance the messages need to travel, characterizes the cost of a location update and search.

In this report, we describe two novel location management strategies: hierarchical and de-centralized, and evaluate them based on their effectiveness in enabling a greater number of successful transmissions between the mobile hosts.

Section 2 discusses some related approaches and Section 3 provides the motivation for this effort. Section 4 presents the Hierarchical Location Management System and Section 5 presents the De-centralized Location Management System. A comparison of both these strategies is discussed in Section 6.

1.1 MOBILE PEER-TO-PEER COMPUTING

The technique used to create of many of the contemporary wireless networks is by forging a relationship between mobile devices (or mobile hosts) and fixed base-stations. The disadvantage of this configuration is that the fixed infrastructure offered by the base-stations limits device mobility and overall network deploy-ability [8]. As a result, these types of networks are not well suited for rapid or temporary network deployment and for environments where it is difficult to achieve adequate base-station coverage. Extensive research is being done in wireless ad-hoc networks. It is now possible to design mobile systems using a peer-to-peer architecture.

Some of the features of ad-hoc networks that apply to P2P systems are:

- **Self-organizing**: Every time a mobile host moves, it needs to re-discover which mobile hosts are reachable. It does this by sending a “ping” message in all directions and listens for corresponding “pong” messages. The strength of the “ping” message weakens as distance increases giving the mobile host a limited range within which “ping” messages can be “heard”. This range is called the *scan range* of the mobile host.
- **Fully decentralized**: No central server exists in a P2P environment. Therefore every mobile host is equally important within the network.
- **Highly dynamic**: The topology of mobile P2P systems can change very rapidly. Therefore within P2P systems, one will find that communication end-points frequently move independently of one another.
- **Low cost**: Wireless ad hoc networks are built from low-cost transceivers and do not incur charges for provider access and airtime.

1.1.1 Peer Discovery & Synchronization

Because of unpredictable mobility of mobile hosts in ad-hoc networks, discovering resources becomes a challenge. Efficient algorithms are therefore required for a peer to detect the presence of nearby peers, share configuration and service information with those peers, and monitor when peers have become unavailable (leave the network or get disconnected). Peers also need to synchronize information about each other that they have gleaned from the various discovery algorithms.

1.1.2 Communication

Mobile hosts are characterized by limitations of power supply and bandwidth, high latency, low availability and low connection stability. In addition to these, when considering the environment of a wireless ad-hoc networks we have two important issues:

- In ad-hoc networks, point-to-point communication is only possible between hosts that are within each other's transmission range.
- The frequent failure and re-activation of links leads to increased network congestion. The network routing algorithm is hence required to react to topology changes [9]. This is because communication between arbitrary peers requires routing over multiple-hop wireless paths whose end-points are likely to be moving independently of one another.

2. RELATED WORK

Numerous strategies have been proposed to solve the problem of communication between two mobile hosts in an ad hoc network. Notable among them are:

1. Flooding – broadcast the data
2. Swamping – connect to all mobile hosts and then broadcast
3. Random pointer jump – connect to random hosts until receipt of data
4. Iterative deepening [4] – iteratively increase depth within which to flood
5. Directed BFS [4] – intelligently select a host to propagate the data
6. Global, pre-computed [6,7,14,16] – maintain tables or complete topology
7. On-demand routing [5] – build routes using some query packets
8. Location based routing [15] – use a location server to determine route

Location information based routing has traditionally used a client-server model. Some of the approaches have used forwarding addresses to keep track of objects that have moved [12, 2]. Yet other methods proposed by [3] or [11] have made use of hierarchical location servers, while [10] and [13] have solved the problem using registry-based approaches, like Location Query Services or caches of location data maintained at the Internet access points.

3. MOTIVATION

1. Existing mobile ad hoc routing algorithms do not explicitly consider the location of a destination node. These algorithms encounter scalability problems as the network size increases, since the amount of information required for making routing decisions increases almost exponentially. In addition, due to host mobility, such approaches would suffer from either outdated information or periodic flooding of updates. If one considers the location information of destinations, the route discovery overhead can be reduced.
2. Each node in the network only needs to know the location of the destination and its neighbor's location to make a forwarding decision. The self-describing nature of location information is the key to achieving a stateless property. For the routing protocols to be useful in larger networks, geographical ad hoc routing protocols are heavily dependent on the existence of scalable location management services, which are able to provide the location of any host at any time throughout the entire network.

4. HIERARCHICAL LOCATION MANAGEMENT

The simplest approach to Location Management is to have a central, well-known entity (*or location server*) manage the location of all the mobile hosts and make them available on-demand. For efficiency reasons, a hierarchy of location servers is maintained, in the form of a tree, instead of a single server. Each location server maintains information regarding the mobile hosts residing in its sub-tree.

In [11], the entities at the leaf level of the tree are base stations. Base stations keep track of the location of a small set of mobile hosts that are within its range. The location servers are informed of changes that take place to the records in their children nodes. Some of the different methods proposed in the paper for propagating the updates to the location servers are: full updates, lazy updates, limited updates, jump updates, no updates, and path compression updates.

Our hierarchical location management system is notably different from the system offered by [11]. The primary reason for this dissimilarity is because we chose to focus on pure ad hoc environments where every node is mobile. Clearly, [11]'s unmovable base stations and location servers were an unsuitable solution in an ad hoc environment. Therefore we devised our own hierarchical location management system in which all entities were mobile. We ensured that the behavior of our system resembled well-known hierarchical location management systems.

4.1 Description of System

In our new Hierarchical Location Management system, a hierarchy of base stations (*or beacons*) is assumed. The beacons are mobile and they additionally function as location servers. Logically, all beacons are nodes of a binary tree. The tree structure is determined *apriori* and all beacons are aware of their position in the tree. Each beacon maintains information about a small set of mobile hosts that can contact it and the mobile hosts that are close to its children.

4.1.1 Mobile Host Registration

When a mobile host detects a beacon within its scan range, it registers itself at the beacon. The beacon adds the new mobile host to its table and informs its parent beacon of the new registration. The parent beacon performs a similar update and informs its parent. This continues until the root beacon is reached or a parent beacon is reached that was aware of the mobile host's previous location. Once registered the mobile host can use that beacon to send or receive data packets.

Mobile hosts that lose contact with their beacon because they have moved too far away re-initiate a search for a new beacon. If a mobile host were successful in locating another beacon that is within its scan range, the mobile host would need to re-register itself with the new beacon. As described, the new beacon would propagate the new registration up the nodes of the tree until the root beacon is reached or a parent beacon is reached that had information about the mobile host's previous location.

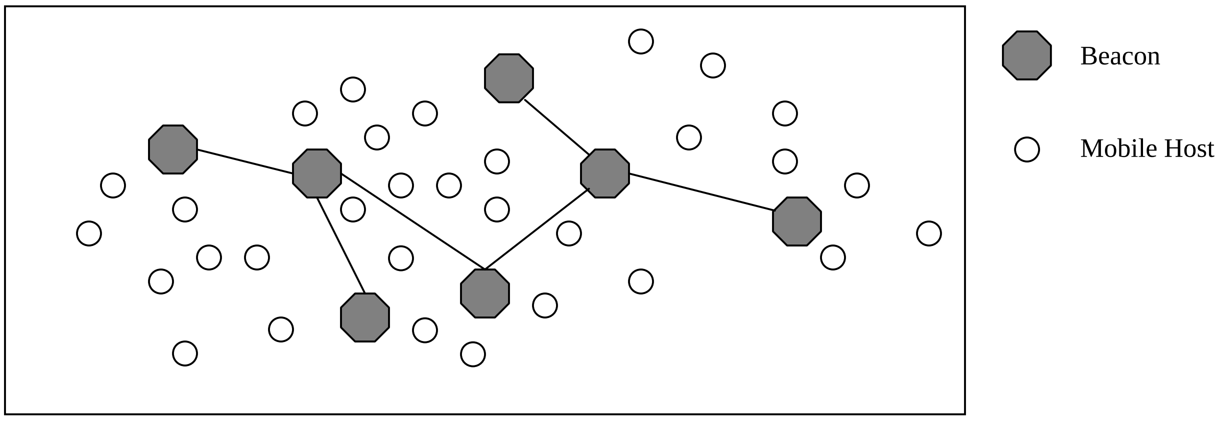


Figure 1: Beacons that form a logical tree move like the mobile hosts that use them for communication.

4.1.2 Mobile Host-to-Mobile Host Communication

Beacons serve to aid the communication between mobile hosts. They are equipped with larger scan ranges, processing power and transmission bandwidth than the mobile hosts. In our implementation of the system, we assume that all beacons can see each other due to their superior scan range. Additionally, we programmed the beacons to have five times the transmission bandwidth of the mobile hosts.

Such a scenario can be imagined to occur on a battlefield where soldiers carrying small mobile devices are the mobile hosts and tanks that are equipped with powerful dish antennas are the beacons. While the soldiers and the tanks move about within the battlefield, the soldiers might want to communicate with each other about their current situation so that the battle is well coordinated.

Returning to our implementation of the hierarchical location management system, we describe how two mobile hosts can communicate. A mobile host (sender) that wants to send some data packets to another mobile host (receiver) will begin by sending it to the beacon within its scan range. If the sender is not near any beacon, it will buffer the data until it finds itself near a beacon. On receipt of the data packets from the sender, the sender's beacon attempts to determine the beacon at which the receiver has registered. By sending a very small control packet, with only the receiver's identity, the request can propagate quickly up the tree until it either reaches the root or a parent that has knowledge about the receiver. If the receiver has not registered at any beacon because none of them are within its scan range, the control packet will reach the root beacon without successfully finding any record about the receiver. This results in a failed transmission and all data packets sent by the sender are dropped by the sender's beacon for lack of route.

However, if a beacon were found that records either the registration of the receiver or records a sub-tree that might contain the receiver's beacon, then the sender's beacon would transmit the data packets through beacons of the tree to the receiver's beacon. There is a possibility that the receiver might have moved away from its beacon into an area not near any beacon. Since only registration updates are propagated through the beacon tree, the sender's beacon or any other beacon would have no way of knowing that the route through the receiver's beacon is actually a dead-end. Therefore, when the receiver's beacon receives the data packets via a beacon-to-beacon transfer through the tree, it drops it due to the unavailability of the mobile host. Only situations where the receiver's beacon can transmit to the receiver are considered successful transmissions.

Notice that previously we mentioned the superior scan range of the beacons. Thus, a beacon that lies outside the scan range of a mobile host could send data to it. The receiving mobile host would not be able to acknowledge this data due to its limitations in scan range. For this reason, we do not permit a beacon to transmit data to a mobile host, unless the beacon lies within the scan range of the mobile host.

4.2 Simulation and Testing

4.2.1 Parameters to consider

For the purpose of empirically determining the effectiveness of our hierarchical location management system, we built a discrete time simulation tool over Windows 2000. Effectiveness was measured using the following two parameters:

1. Percentage of successful transmission: This is the percentage of all the transmissions that successfully reached receiving mobile hosts from amongst all the attempted transmissions by the sending mobile hosts.
2. Percentage of partially successful transmissions: This is the percentage of all the transmissions that were successfully sent from the sending mobile host to its beacon.

Subtracting parameter 1 from parameter 2 will give the percentage of data packets that were dropped by the nodes of the beacon tree. Additionally, by keeping a count of the number of control packet transmissions required by the beacons for each data packet transmitted, the average control packet overhead can be calculated.

4.2.2 Simulator Setup

We ran our experiments on our discrete time simulator with the following arrangement:

1. A grid 50 units by 50 units within which the mobile hosts and beacons move.
2. 150 mobile hosts with varying scan ranges. For simplicity, scan ranges are not assumed to be circles with the mobile host at the center. Instead, the scan range is a square with the mobile host at the center. Scan ranges vary from a square, 1 unit by 1 unit to a square, 50 units by 50 units.
3. 15 beacons that are logically connected like a tree. Beacons can reach any point within the grid because of their superior transmission range.
4. All entities (mobile hosts and beacons) within the grid can move using a random waypoint mobility pattern. The initial position of the entity is random. Three schedules are planned for each entity. This means that three random positions are picked as possible destinations for the entity. In addition, three randomly picked values from the set {0,1} will each act as the mask associated with a schedule. If the mask for a schedule is 1, the entity will move to the destination for that schedule

within an amount of time randomly picked from the set {1, 2, 3, 4, 5}. Else, if the mask was 0 for a particular schedule, the entity will remain at its current location for an amount of time randomly picked from the set {1, 2, 3, 4, 5}.

5. For data transmission, each mobile host randomly decides whether to transmit only one data packet, two consecutive data packets or no data packet. The receiving mobile host is picked randomly as is the time slot in which to begin the transmission attempt.
6. The mobile host has an approximate uplink and downlink capacity of 2Mbps. The data packets are 1.5 Mbps each. This means that a mobile host can only send or receive a single data packet in one time slot.
7. Beacons have five times the uplink and downlink capacity of the mobile hosts. Therefore, a data packet transmitted by a beacon can go through five different beacons in one time slot.
8. Control packets are very small and do not cause any significant increase in data traffic.
9. Beacons have unlimited buffer capacity and therefore no data packets are lost or delayed due to buffer overflows. This attribute can be changed to allow limited buffers for the beacons. However, due to lack of time we did not run tests for this case.

4.2.3 Running the Simulator

There are three input files that are used to set up the simulation environment:

1. `beacon.in` – provides the number of beacons (has to be 2^n-1 for it to be a binary tree hierarchy), lists each beacon along with the identity of its parent, its initial position and the 3 mobility schedules with their masks, new positions and velocity.
2. `nodes.in` – provides the number of mobile hosts, lists the mobile host numbers, and for each mobile host it lists the initial position, the three possible destinations, the three mobility masks and the velocity.
3. `cbr.in` – provides the number of mobile hosts, the size of a data packet in bits, the bandwidth available to a beacon, and for each mobile host it lists a mask which

indicates if a mobile host would transmit or not, the number of data packets which it might transmit, the time slot at which an attempt will be made to transmit the data packet and the destination mobile host.

The simulator code has the following constant integers that can be changed for different arrangements. The Microsoft Visual Studio compiler can be used to re-build the executable.

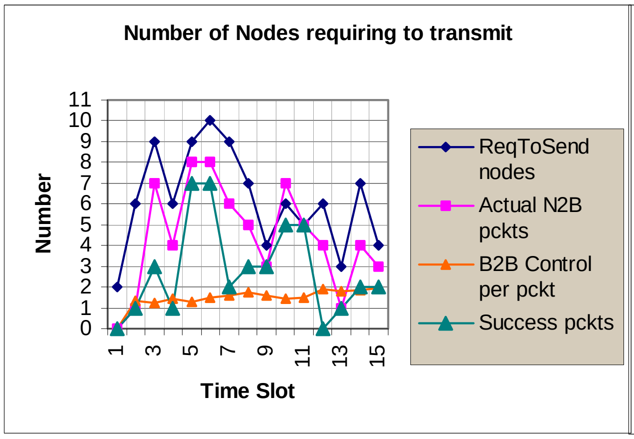
```
GRID_X = 50;     // size of Grid in X-direction
GRID_Y = 50;     // size of Grid in Y-direction
SCAN_RANGE = 7;  // mobile host scan range is now 7x7
NBUFFSIZE = 50;  // size of buffer on the mobile host
BBUFFSIZE = 50;  // size of buffer on the beacon
```

A log file named "`log.out`" is created in which attempts by mobile hosts to transmit, successful mobile host to beacon transmissions, dropped data packets, etc. are logged for later tabulation.

4.2.4 Simulation Results

Due to lack of space we only display two results of our tests. All results have been averaged over two mobility patterns. The graph plots four line-curves that are described below. On the X-axis is time:

1. **ReqToSend nodes** – this line displays the number of mobile hosts that are attempting to transmit in a particular time slot.
2. **Actual N2B pckts –** this line displays the number of data packets that are successfully sent to beacons from mobile hosts attempting to transmit in that time slot.
3. **B2B Control per pckt –** this line displays the control packet overhead for that time slot. It is calculated as a ratio of the total number of control packets transmitted to the total number of data packets transmitted within that time slot. Total data packet transmissions include mobile host-to-beacon or beacon-to-beacon data packet transfers.
4. **Success pckts –** this line displays the total number of data packets that make it to the receiving mobile host in a particular time slot.



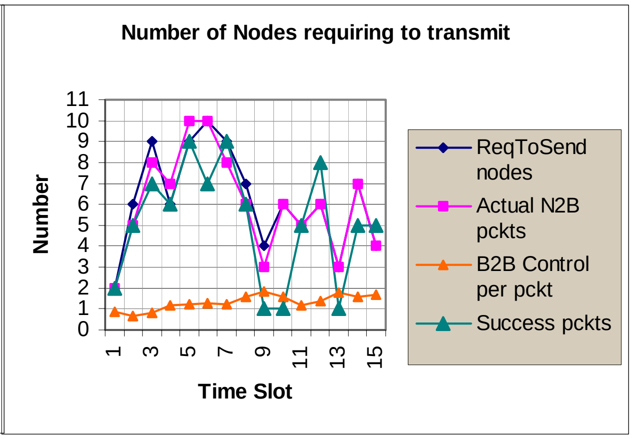
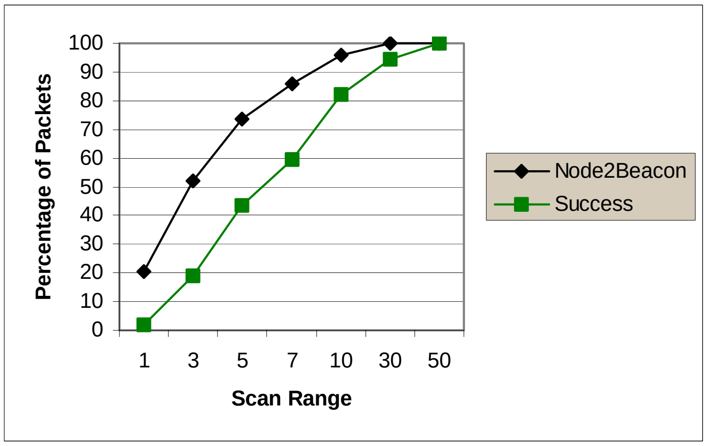


Figure 2: Mobile host scan range, 5 units by 5 units

Figure 3: Mobile host scan range, 10 units by 10 units

Note how the "B2B Control per pckt" curve remains almost constant irrespective of the number of attempted transmissions in both the graphs above. Also noticeable is the increase in the number of successful transmissions and partially successful transmissions from scan range 5x5 to scan range 10x10. The obvious reason for this is because of the increase in the scan range for a mobile host.



- **Node2Beacon –** this is the percentage of partially successful transmissions
- **Success –** this is the percentage of completely successful transmissions

Control packet overhead = 1.27 control per data

Figure 4: Effectiveness of the system.

5. DE-CENTRALIZED LOCATION MANAGEMENT

The Local Indices technique [4] is one of the efficient search techniques designed for searches within a peer-to-peer network. In this technique, a node *n* maintains a data structure that records the location information of all nodes adjacent to it and within *r* hops away from itself, where *r* is defined as the radius of the index. The resulting data structure is termed as the Local Indices for the node. When a node receives a data packet for routing, it checks the Local Indices to determine if the destination node is within *r* hops of itself. If so, it would forward the packet to the next adjacent node to efficiently route it to the destination. Else if the Local Indices does not record the destination node, the current node would forward the packet to another node that exists *r* hops away from it.

The Local Indices technique causes an increase in the amount of data stored at a mobile host. In some cases this might be a drawback, because mobile hosts have limited data storage capability and might not be willing to store large amounts of data. Using a smaller value for *r* can minimize the amount of data stored at each mobile host. The storage of information can be very advantageous, for the reason that by querying a small number of mobile hosts we can get location information of many mobile hosts.

The Local Indices algorithm uses the following system-wide ranges:

- **Scan Range:** See Section 1.1 for description of this constant value.
- **POLICY:** This is the same as the radius *r* described above.

In peer-to-peer mobile ad-hoc networks, the mobile hosts are constantly moving. Therefore data structures that provide location information about other mobile hosts need to be updated at regular intervals. This is an overhead for each node. Additionally, whenever a mobile host joins or leaves the network, extra computation is needed to create and maintain the indices at each mobile host.

When a mobile host joins a network, it scans its neighbors within a depth of *r* hops from it, and forms Local Indices by requesting their identities. During the process of formation of the Local Indices, each mobile host that received the scan will update its own Local Indices with a record of the mobile host that recently joined the network.

Periodically, every mobile host sends a *ping* message to all mobile hosts in its Local Indices. In response, every mobile host that receives a *ping* message responds with a *pong* message, indicating its existence. If a mobile host does not receive a pong message from a mobile host for a fixed interval of time, it assumes that the mobile host is no longer within its radius or is currently disconnected. A corresponding change is made to the Local Indices data structure.

5.1 Description of System

5.1.1 Building the data structures

We assume a random placement of all mobile hosts within a region. Once the mobile hosts have been placed, we form an adjacency list and Local Indices at each mobile host following these steps:

1. We scan the scan range for all directly reachable mobile hosts. Using a spiral matrix traversal that starts at the current mobile host, which is treated as the center of the matrix, and following the path of the spiral until the scan range is reached, the algorithm for the scan is implemented.
2. The scan helps build an adjacency list of mobile hosts that can be directly reached from the current mobile host.
3. Next we need to construct the Local Indices data structure. This data structure is built using the adjacency list just created.
4. The Local Indices data structure is formed by retrieving information from each of the mobile hosts present in the adjacency list. This is a recursive process because each mobile host has its own adjacency list. Therefore we set the limit on the recursion. This limit also restricts the depth to which we need to build the Local Indices. The system-wide constant POLICY is used for this very purpose.
5. Once the Local Indices have been built at each mobile host, the network is ready to send and receive data packets. The Local Indices are rebuilt periodically because mobile hosts occasionally leave the network or get disconnected.

5.1.2 Communication

The process of maintaining data structures is synchronized with the sending or receiving of data packets. We follow the following steps in routing packets:

1. Data packets that need to be routed from any mobile host are done so based on the Local Indices at that mobile host.
2. The mobile host checks for a record of the receiving mobile host in its local indices. If it finds it, the mobile host immediately forwards the packet to one of its adjacent mobile host so that the packet can be routed to the receiving mobile host. Else, if the receiving mobile host does not exist, then the following action is taken by the current mobile host:

- It temporarily delays the packet for a fixed time and then later retries the send to the receiving mobile host.
- However, if the receiving mobile host is still not reachable within the time assigned, the sending mobile host forwards the data packet to a mobile host that has a higher probability of reaching the receiving mobile host. We assume that this probability value is a function of the size of a mobile host's Local Indices and its packet delivery history (to see how active the mobile host performed in the past).

3. If the data packet does not reach the receiving mobile host in a fixed amount of time, we treat the transmission as *FAIL*. The packet is assumed to be dropped in one of the following cases:
    - If the transmission on a data packet has not yet begun and the *TIME_TO_KILL* value has passed.
    - If the data packet has been traveling around the network and has not yet reached the receiving mobile host and the *TIME_TO_KILL* value has passed.
4. If the packets reach the destined mobile hosts in a fixed amount of time, we treat it as a *SUCCESS*.

5.2 Simulation and Testing

5.2.1 Parameters to consider

For the purpose of empirically determining the effectiveness of our hierarchical location management system, we built a discrete time simulation tool over Windows 2000 that was identical to the one built to evaluate the hierarchical location management strategy. Effectiveness was measured by the percentage of successful transmission. This is the percentage of all the transmissions that successfully reached receiving mobile hosts from amongst all the attempted transmissions by the sending mobile hosts. Additionally, by keeping a count of the number of control packet transmissions required by the mobile hosts to maintain the data structures gives us the control packet overhead of this strategy.

5.2.2 Simulator Setup

The simulator for this strategy was setup with exactly the same arrangement as that of the hierarchical location management system. Even the same randomly calculated values for mobility and data packet transmission were used to ensure a fair comparison. The only difference was that in this strategy, beacons were absent. See section 4.2.2 for a description of the simulator setup.

5.2.3 Running the Simulator

There are two input files that are used to set up the simulation environment:

1. `nodes.in` – provides the number of mobile hosts, lists the mobile host numbers, and for each mobile host it lists the initial position, the three possible destinations, the three mobility masks and the velocity.

2. `cbr.in` – provides the number of mobile hosts, the size of a data packet in bits, and for each mobile host it lists a mask which indicates if a mobile host would transmit or not, the number of data packets which it might transmit, the time slot at which an attempt will be made to transmit the data packet and the destination mobile host.

The simulator code has the following constant integers that can be changed for different arrangements. The Microsoft Visual Studio compiler can be used to re-build the executable.

```
GRID_X = 50;     // size of Grid in X-direction
GRID_Y = 50;     // size of Grid in Y-direction
SCAN_RANGE = 7;  // mobile host scan range is now 7x7
POLICY = 2;      // the value of the system-wide constant
NBUFFSIZE = 50;  // size of buffer on the mobile host
BBUFFSIZE = 50;  // size of buffer on the beacon
```

A log file named "`log.out`" is created in which attempts by mobile hosts to transmit, successful mobile host transmissions, and dropped data packets, etc. are logged for later tabulation.

5.2.4 Simulation Results

In our experiments, we vary the mobile host scan range and the system-wide constant of POLICY. Due to lack of space we only display some of the results of our tests. All results have been averaged over two mobility patterns. The graph plots four line-curves that are described below. On the X-axis is time:

1. **ReqToSend nodes –** this line displays the number of mobile hosts that are attempting to transmit in a particular time slot.

2. **Actual N2N pckts –** this line displays the number of data packets that are successfully transferred between two mobile hosts in a particular time slot while attempting to route the data packet to its intended destination.

3. **Control per node –** this line displays the control packets per node in a particular time slot. It represents the control packet overhead of this location management strategy.

4. **Success pckts** – this line displays the total number of data packets that make it to the receiving mobile host in a particular time slot.

5.2.4.1 POLICY is kept constant and the scan range is varied.

In this case, we consider the POLICY to be fixed at 2. The scan range is varied from 1x1 up to 50x50. Increasing the scan range means that we are allowing a mobile host to contact more of its neighbors in a single hop.

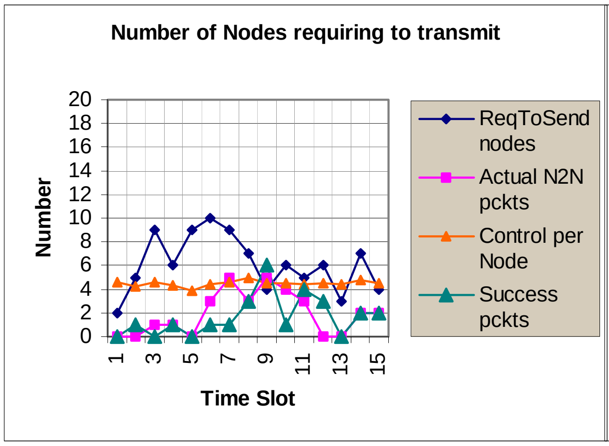


Figure 5: Mobile host scan range, 5 units by 5 units and POLICY=2

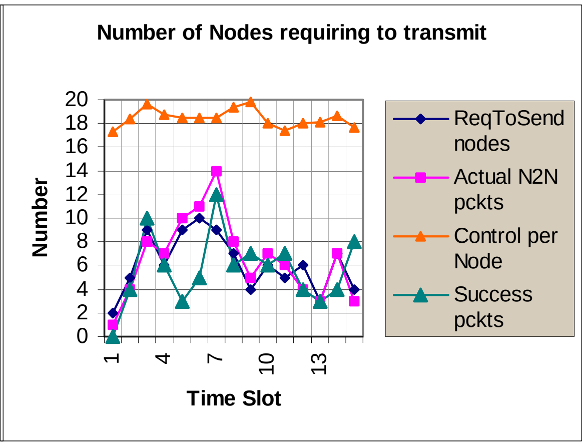


Figure 6: Mobile host scan range, 10 units by 10 units and POLICY=2

Note the following conclusions based on the graphs above:

1. Number of control packets drastically increases with the increase in scan range. This is because of an increase in number of mobile hosts that can now be added to the adjacency list, therefore resulting in an increase in size of Local Indices.
2. The success rate increases from 27% to 92%, showing that increasing the range of a mobile host increases the success rate of delivering packets.
3. The "Actual N2N pckts" line shows more mobile host-to-mobile host transmissions due to the increase in the size of Local Indices at each mobile host.

5.2.4.2 Scan Range is kept constant and POLICY is varied.

In this case, we fix the value of the scan range at 5x5. The POLICY variable is varied from 2 up to 5. Increasing POLICY only enlarges the Local Indices. It does not increase the number of mobile hosts that can be contacted in a single hop like in section 5.2.4.1.

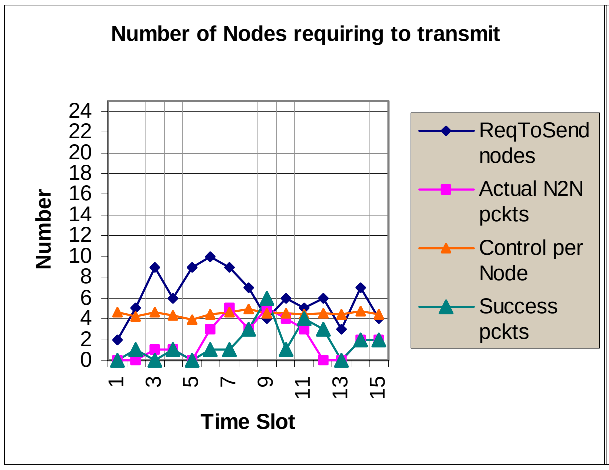


Figure 7: Mobile host scan range, 5 units by 5 units and POLICY=2

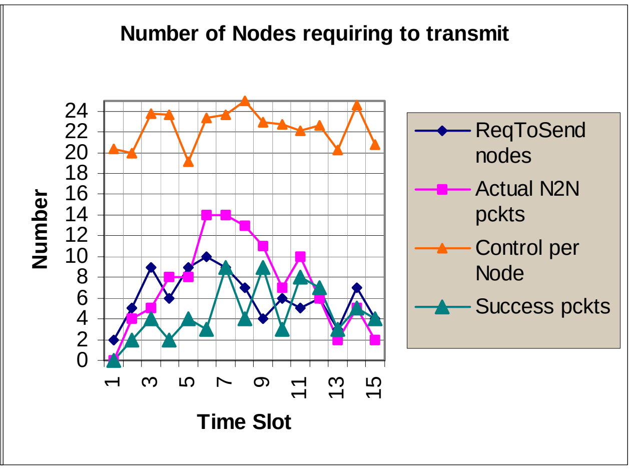


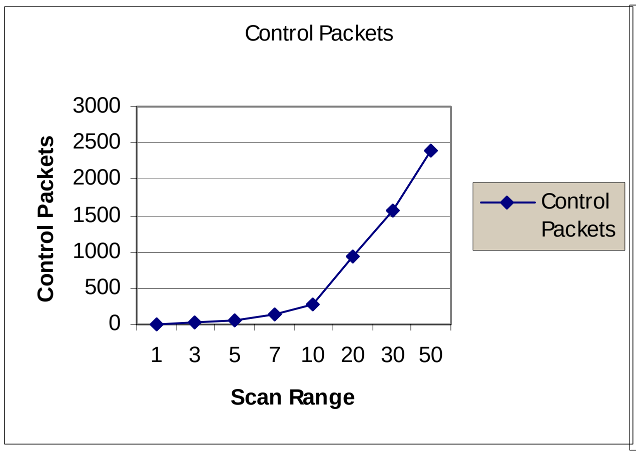
Figure 8: Mobile host scan range, 10 units by 10 units and POLICY=3

Note the following conclusions based on the graphs above:

1. The number of control packets drastically increases with the increase in policy. This is because the size of Local Indices increases.

2. The success rate increases from 27% to 73%, demonstrating that increasing the policy of a mobile host increases the success rate of data packets transmissions.

3. The “Actual N2N pckts” line shows more mobile host-to-mobile host transmissions due to the increase in the size of Local Indices at each mobile host.



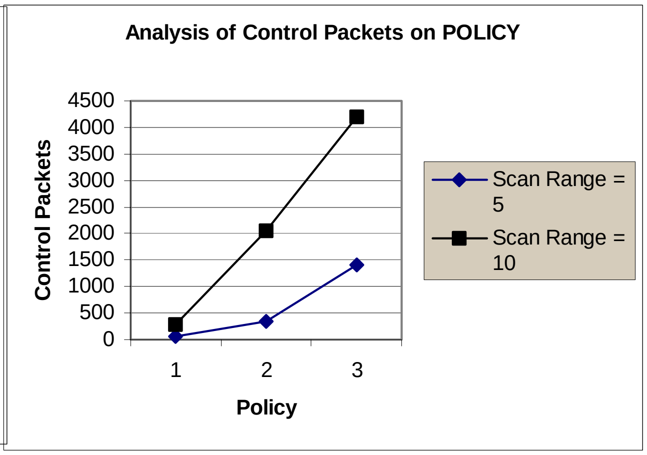


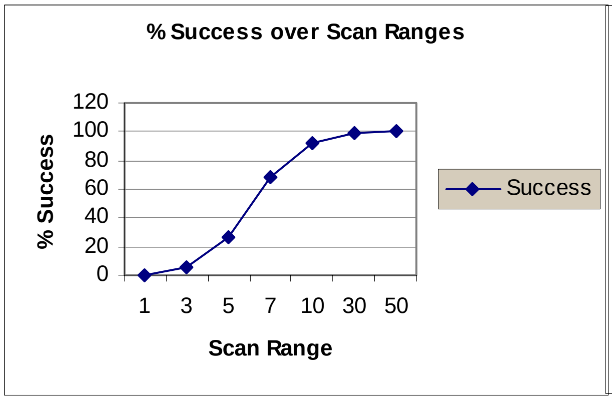
Figure 9: Analysis of Control Packets over Scan Range

Figure 10: Analysis of Control Packets over POLICY

Comparing both the cases above, we could draw the following conclusions:

1. The number of control packets generated by increasing the policy is greater than the number generated by increasing the scan range of mobile hosts.
2. The number of mobile host-to-mobile host transmissions increases more with an increase in policy than mobile host scan range. This is due to an enlargement of the Local Indices data structure that occurs with an increase in policy. This increases the number of mobile hosts that are visible to a single mobile host.

The graph below shows the effect on success rate in both the cases.



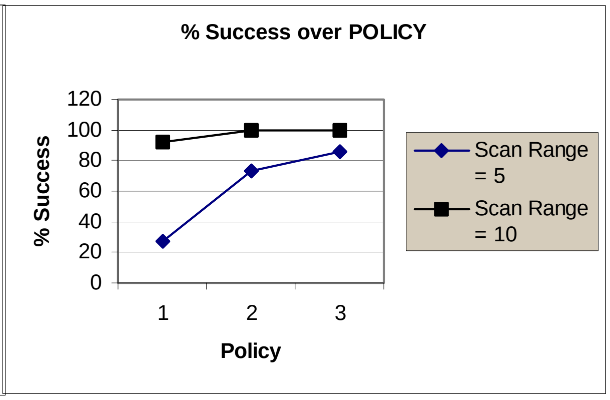


Figure 11: % Success over variations in scan range and POLICY=2

Figure 12: % Success over variations in POLICY with two scan ranges

6. FINAL COMPARISONS

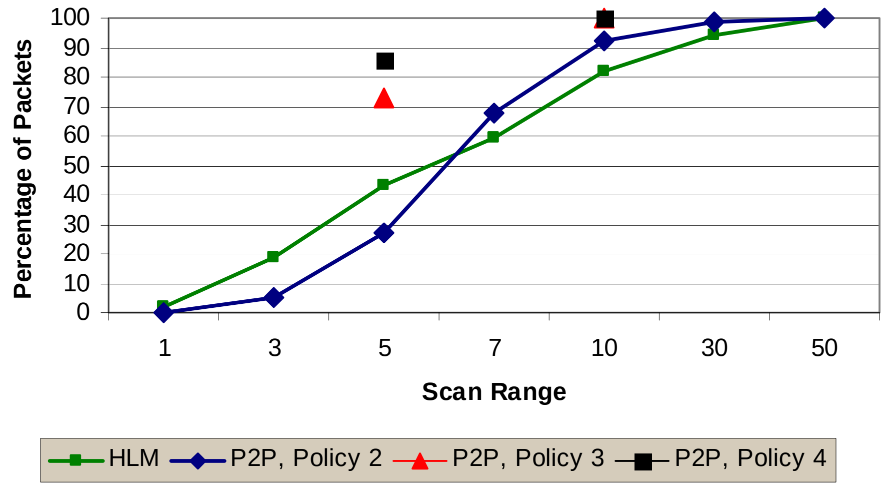


Figure 13: Percentage successful transmissions.

HLM: Hierarchical Location Management, P2P: Peer-to-Peer Location Management

The above graph is a combination of Figures 4 and 11 and compares the percentage of successful transmissions between the two location management strategies. These results have been averaged over two different mobility patterns. The graph demonstrates the superiority of the de-centralized location management strategy at higher mobile host scan ranges. If the POLICY were to be increased, the de-centralized location management strategy manages to achieve better successful transmissions at even lower scan ranges. However, one thing must be pointed out at this stage. In the hierarchical location management strategy the number of control packet overhead is not only very small; it is almost constant. This cannot be said of the de-centralized approach in which the control packet overhead increases exponentially as the policy or the scan range increases. Some might argue that control packet overhead is not a source of concern because of the small size of the control packets. However with an exponential increase, the total amount of bandwidth consumed in control packet overhead might play an important role in determining which strategy works best for a particular situation.

7. REFERENCES

[1] A. Rowstron and P. Druschel, “Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility”, *ACM Symposium on Operating Systems Principles*, Canada, 2001.

[2] B. Awerbuch and D. Peleg. “Concurrent online tracking of mobile users.”, *ACM SIGCOMM Symposium,* October 1991.

[3] B. R. Badrinath, T. Imielinski and A. Virmani. “Locating Strategies for Personal Communicating Networks.”, *Proceedings of the IEEE GLOBECOM Workshop on networking of Personal Communication,* December 1992.

[4] Beverly Yang, Hector Garcia-Molina, “Efficient Search in Peer-to-Peer Networks”, Computer Science Department, Stanford University, 2001.

[5] C. E. Perkins and E. M. Royer, “Ad hoc on demand distance vector (AODV) routing (Internet Draft)”, Aug. 1998.

[6] D. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc networks”, in *Mobile Computing* (T. Imielinski and H. Korth, eds.), Kluwere Academic Publishers, 1996.

[7] D. Johnson, D. A. Maltz and J. Broch, “The dynamic source routing protocol for mobile ad hoc networks (Internet-Draft)”, Mar. 1998.

[8] Gred Kortuem, “Proem: A Peer-to-Peer Computing Platform for Mobile Ad-hoc Networks”, Department of Computer Science, University of Oregon.

[9] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. ACM Trans. on Computer Systems, 10(1): 3, February 1992

[10] M. Spreitzer and M. Theimer, “Providing Location Information in a Ubiquitous Computing Environment.”, *Technical Report,* Xerox PARC, 1993.

[11] P. Krishna, N. H. Vaidya, D. K. Pradhan, “Static and Dynamic Location Management in Distributed Mobile Environments”, Technical Report 94-030, Department of Computer Science, Texas A&M University, June 1994.

[12] R. J. Fowler. “The Complexity of using Forwarding Addresses for Decentralized Object Finding.”, *ACM SIGCOMM Symposium*, 1986.

[13] S. F. Wu and Charles Perkins. “Caching Location Data in Mobile Networking”, *IEEE Workshop on Advances in Parallel and Distributed Systems,* October 1993.

[14] V. D. Park and M. S. Corson, “Temporally-ordered routing algorithm (TORA) version 1 functional specification (Internet Draft)”, Aug. 1998.

[15] Young-Bae Ko, Nitin H. Vaidya: Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. MOBICOM 1998: 66-75

[16] Z. J. Haas and M. R. Pearlman, "The zone routing protocol (ZRP) for ad hoc networks (Internet Draft)", Aug. 1998.