CLOCK SYNCHRONIZATION FOR MOBILE AD HOC NETWORKS

A Senior Honors Thesis

by

RAJAN CHANDRA

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A & M University
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

April 2003

Group: Engineering and Physics 1

CLOCK SYNCHRONIZATION FOR MOBILE AD HOC NETWORKS

A Senior Honors Thesis

by

RAJAN CHANDRA

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A & M University
in partial fulfillment for the designation of

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOW

Approved as to style and content by:

[illegible]
Jennifer L. Welch
(Fellows Advisor)

[illegible]
Edward A. Funkhouser
(Executive Director)

April 2003

Group: Engineering and Physics 1

ABSTRACT

Clock Synchronization in

Mobile Ad Hoc Networks. (April 2003)

Rajan Chandra
Department of Electrical Engineering
Texas A&M University

Fellows Advisor: Dr. Jennifer L. Welch
Department of Computer Science

As mobile networking advances, there is a need for services such as clock synchronization that improve performance and support the development of higher-level applications. This can be achieved by adapting existing algorithms (such as the Network Time Protocol) used in wired networks (e.g. Internet) to Mobile Ad Hoc Networks (MANets). It may also be achieved by developing other algorithms that achieve clock synchronization and may be suitable for MANets. Using the Network Time Protocol (NTP) as a basis, an algorithm was developed for clock synchronization in Mobile Ad Hoc Networks. NTP is the Internet standard for clock synchronization and has been designed for wired networks. Since mobile ad hoc networks are inherently different from wired networks, which are static, several components of NTP were analyzed and modified in developing the algorithm for MANets. Simulations were performed for testing a basic version of the algorithm using Network Simulator 2, a discrete event simulator widely used in research for simulating mobile ad hoc networks. Simulation results reveal interesting information about how the tested algorithm performed. Possible ways to improve the algorithm are also discussed. Another approach to achieve clock synchronization uses MANet specific communication primitives with Minimum Connected Dominating Set (MCDS) approximation algorithms. This involves using a subset of nodes in the network to broadcast clock information to neighboring nodes. Several MCDS approximation algorithms have been developed and it is important to analyze their strengths to determine how suitable they are for clock synchronization in mobile ad hoc networks. A few such algorithms are discussed. This work may encourage further research in improving the proposed algorithm or in the development of NTP based clock synchronization algorithms. Finally, it may contribute to the implementation and advancement of clock synchronization in mobile ad hoc networks.

ACKNOWLEDGMENTS

I am very grateful to my advisor, Dr. Jennifer L Welch, Department of Computer Science, Texas A&M University for being an excellent mentor during the research fellows program. She met with me once a week, during which she monitored my progress, solved my difficulties and guided me on future work. Apart from this, she spent valuable time advising me on graduate school and career planning. Dr. Welch also took keen interest in the research program and attended events and presentations whenever she was in town, which was very encouraging.

I would also like to thank Guangtong Cao, a PhD student of Dr. Welch. He helped me significantly in my research, especially with the Network Simulator 2 simulations.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The last ten years have seen a rapid evolution in technology. Computing is becoming more pervasive and distributed and networks are becoming ubiquitous in the modern world. One of the most promising fields within networking involves mobile ad hoc networks. A mobile ad hoc network [1, 2, 5] is a network where nodes communicate by sending messages over wireless links. A node in these networks is an element of a network and can be computers, devices, automobiles, etc. Ad hoc networks are characterized by frequently changing topology of the network; in other words, the communicating nodes are often moving with respect to their relative positions. Another characteristic of mobile ad hoc networks is that they require no preexisting infrastructure to operate. Thus, they differ from many other networks such as cellular phone networks that are wireless and have mobile nodes, but need infrastructure such as communication towers. The attributes of mobile ad hoc networks: mobility of nodes, wireless communication links between nodes and no preexisting infrastructure requirement, makes them very useful and versatile. As networking needs and applications expand, mobile ad hoc networks will be one of the fastest growing network types.

While tremendous research has been done in the field of ad hoc networking, most of it has focused on the design of routing and medium access control protocols. In simple terms, these components of a network are responsible for the transfer of data packets or messages from one node to another. They facilitate communication between nodes. Only recently has the research focus shifted to also include the development of distributed services for mobile ad hoc networks. The inclusion of these services in mobile ad hoc networks can be beneficial to several other applications. One such service is clock synchronization, which plays an important role in traditional networks, especially for applications such as stock trading, messaging, file transfer and air traffic control. It is equally significant in mobile ad hoc networks.

Nodes in a network often possess individual hardware clocks that may not be very accurate. They therefore may tend to drift significantly away from real time. Clock synchronization refers to the maintaining of logical clocks in an effort to keep them closer to real time than the actual hardware clocks. It includes several processes that maintain logical clocks by computing adjustments to the physical clocks [8].

Definitions

Computer Networks [4]: A network is a collection of devices that are connected to each other by links using which these devices can communicate with each other. Nodes are the elements of a network and can consist of computers, devices, etc. The Internet, for example, is a worldwide network of computers as nodes. A topology of a network describes which nodes can communicate directly with other nodes.

Mobile Ad hoc Networks (MANets): A mobile ad hoc network [1, 2, 5] is a network where nodes communicate by sending messages over either a direct link or a sequence of wireless links, without the need for any pre-existing infrastructure. Nodes are the elements of a network and can be computers, devices, etc. Ad hoc networks are characterized by frequently changing topology of the network; in other words, the communicating nodes are often moving with respect to their relative positions.

Clock Synchronization: Each node or processor, p_i, has a hardware clock with time $HC_i(t)$ [3,9]. A logical clock with time $LC_i(t)$ is maintained such that $LC_i(t) = HC_i(t) + Adj_i(t)$, where t is the real time, $LC_i(t)$ is the logical time at node i, $HC_i(t)$ is the hardware clock time at node i, and $Adj_i(t)$ is the value of the adjustment register at node i. In this way, clock synchronization processes attempt to set the logical time at a node close to the real time.

Offset: At any given time, t, the logical time at p_i, $LC_i(t)$ will differ from the real time, t, by an offset. Therefore, in terms of the offset and the real time, the logical clock time can be represented as $LC_i(t) = t + e_i$, where e_i is the offset between the logical clock time and real time. Clock synchronization processes seek to make e_i as close to zero as possible, so that the logical time at a node is close to the real time. The clock synchronization methods discussed in this paper seek to achieve clock synchronization by approximating what the offset, e_i, is for every node, p_i. In testing performed in this project, e_i will be used to determine the performance of the clock synchronization algorithm. A lower offset will be interpreted as better clock synchronization. The algorithm will be assessed by simulating it on test networks (using a network simulator) and calculating the average offset, e_{avg}, of all the nodes in the network.

Minimum Connected Dominating Set (MCDS): A dominating set is a subset D of the nodes in a network G, such that every node, p_i, of the network G is either in the set D or a neighbor of a node in D [11]. A dominating set is connected if the subgraph D is connected. A minimum connected dominating set is a connected dominating set, D with the smallest number of nodes possible in D.

RELATED WORK

Clock synchronization is an important service for networks and provides service to a host of applications ranging from file sharing to network security and protection [7]. Extensive research has been done in development of clock synchronization algorithms and protocols for wired networks such as the Internet. This includes the Network Time Protocol, which is the Internet standard for clock synchronization [3]. However, little work has been done for clock synchronization in mobile ad hoc networks. The inherent differences in mobile ad hoc networks when compared with traditional wired networks suggest a need for different algorithms for mobile ad hoc networks.

Although this problem has been recently addressed by papers such as Römer's [12]), more promising work is required as Römer, in his paper, discusses time transformation functions instead of achieving clock synchronization. Therefore, there may be a need for new clock synchronization algorithms specifically developed for mobile ad hoc networks. Alternatively, algorithms may be adapted from existing ones such as the Network Time Protocol. They may also use MANet specific communication primitives and Minimum Connected Dominating Set (MCDS) approximation algorithms [18]. Extensive work has been done in these fields separately, which can be applied to the problem of clock synchronization [11, 13, 14, 15, 16]. [11] discusses using the broadcasting property of mobile ad hoc networks for clock synchronization. However, the scope of this paper is limited to single hop networks and it does not address multi-hop networks. Also, [13], [14], [15], [16] propose MCDS approximation algorithms that have different characteristics. These characteristics may include time complexity and performance, which refers to how good the algorithm is in approximating a minimum connected dominating set. Papers such as [18] by Cao and Welch, that use MANet specific communication primitives and Minimum Connected Dominating Set (MCDS) approximation algorithms to for clock synchronization are promising and show increasing interest in the problem of clock synchronization for mobile ad hoc networks.

PROBLEM

This project involved investigating some suitable algorithms for clock synchronization in mobile ad hoc networks. These algorithms included versions of NTP (Network Time Protocol) adapted for MANets and using MANet specific communication primitives with Minimum Connected Dominating Set (MCDS) approximation algorithms. In particular the aims of this project were:

- Developing an algorithm based on NTP (Network Time Protocol) for mobile ad hoc networks. NTP is the Internet standard for clock synchronization and therefore designed for wired networks. In adapting it to MANets, several features of NTP have to be considered and possibly modified.
- Analyzing a promising version of NTP testing it on network simulator 2. Results obtained from such testing will be useful in assessing the developed algorithm and in the further development of algorithms for clock synchronization in MANets.
- Exploring clock synchronization algorithms that use MANet specific communication primitives with Minimum Connected Dominating Set (MCDS) approximation algorithms. The broadcasting property of MANets may hold certain advantages in clock synchronization by potentially minimizing the effect of variable message delays. MCDS approximation algorithms may be needed to complement a broadcasting algorithm. This project involved exploring existing MCDS approximation algorithms and analyzing them based on expected strengths and weaknesses.

ADAPTING NTP

The Network Time Protocol (NTP) was developed in the mid 1980s to achieve clock synchronization in wired networks. NTP has advanced significantly in the last fifteen years and is the standard for clock synchronization on the Internet today [3]. It organizes and maintains a set of time servers and transmission paths as a smaller part of a network, or subnet. In NTP, time servers (nodes that provide other nodes on the network with the time) are classified depending on how accurate and reliable their source of time is. Primary servers synchronize directly to external reference sources such as a GPS (Global Positioning System, a satellite based navigation system that provides highly accurate location and time information) or other services such as those provided by national governments and agencies [10]. Primary servers provide time to secondary servers, which in turn synchronize clocks at the lower levels of the network in a tree-like manner. NTP uses basic Internet protocols to function. It operates in two modes: a client-server mode and a peer mode.

At the heart of NTP are the algorithms that synchronize clocks on the Internet. These chiefly comprise the clock filter, clock selection and clock discipline algorithms.

Clock filter and Clock selection algorithms: A node synchronizes with many servers or peers to obtain accurate time information. In turn, it may receive many messages from each of these nodes, periodically. To synchronize its own clock, it has to determine its best estimate of the real time from all the messages it receives. For this, it has to select the best and most reliable message or group of messages and synchronize its clock based on these messages. The clock filter algorithm selects the best estimate of the real time from among all the messages from a single node. The data in an NTP message contains four timestamps, or data fields that store a time (figure 1). The message originates at the client (or the first peer), which includes the local node time at which it is sending the message, T1. When this message arrives at the server, it time stamps it, T2. The server returns the message after updating the information on the message at server time, T3. When the client (or original sender of the message) receives the message, it records its time, T4. Using these timestamps, the node can determine the offset (time by which the local node time lags the server time) and the roundtrip delay (time taken to transmit message from client to server and from the server back to the client). The clock selection algorithm helps determine the real time using the output of the clock filter algorithm.

In developing an NTP based algorithm for MANets, the following features of NTP had to be considered. [Have to improve explanation and elaborate. Add text to explain diagram]

1. Message delays:

Message delay estimation is central to NTP. NTP messages have timestamps of the local times at the client and server when they are received or sent. NTP uses these to determine the total roundtrip time it takes for a message from the time it is sent by the client to the time that the client

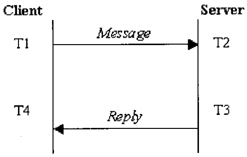

Figure 1: NTP message timestamps. Shows two nodes, A is the client and B is the server. A sends a synchronization message to B at T1. B receives it at T2 and returns it at T3. A gets B's reply at T4,

receives the reply. Figure [number] shows an NTP message and the corresponding timestamps. Therefore the total roundtrip time is

$$\text{Roundtrip Time} = (T4 - T1) - (T3 - T2).$$

The message delay that has to be estimated is the time that it takes the message to reach the client after the server sends it. NTP uses roundtrip time to place an upper bound on the message delay and for its clock filter and clock selection algorithms. It may be important to investigate if message delays in MANets have different characteristics than wired networks. If message delays are expected to be significant, it will be beneficial to incorporate message delay estimation in an algorithm. If message delays are expected to be small, the additional overhead required for delay calculations may not be worthwhile.

2. Offset Estimation:

As described earlier, clock synchronization algorithms may maintain a logical clock at every node, p_i in the network. Also, at any given time, t, the logical time at p_i, $LC_i(t)$ will be away from the real time, t by an offset. A smaller offset implies a smaller difference between the logical clock at p_i and the real time and therefore results in better clock synchronization. NTP estimates this offset in the logical clock of the client based on the following approximation:

$$\text{Offset} = \frac{(T3 - T4) - (T2 - T1)}{2} \quad \text{(where T1, T2, T3, T4 are shown in Fig 1)}$$

This equation can be used to calculate the offset in the algorithm adapted from NTP. It is important to note that the value of the offset calculated will only be an estimate of the true offset. The relation between the estimated offset and the true offset can be found by the following mathematical discussion:

Consider Figure 1. Let the two nodes shown be A and B, where A is the client node and B is the server. Also,

- *T1* be the local time at A (client) when it sends a message.
- *T2* be the local time at B (server) when it receives the message.
- *T3* be the local time at B (server) when it returns the message to A.
- *T4* be the local time at A (client) when it receives the message.
- Let *P1* be the time it takes the message to get from A to B and *P2* be the time it takes the message to get from B to A.
- Let *b* be the time it takes node B to reply to the message, ie, the time between B receiving the message T2 and it sending the message T3.
- *e* be the true offset of clock A or the amount of time by which A leads B. Therefore e= time at A – time at B.

Now at a particular instant when the client A sends out a message, let the server time be t. Then:

$$T1 = t + e$$
$$T2 = t + p1$$
$$T3 = t + p1 + b$$
$$T4 = t + p1 + b + p2 + e$$

Substituting these into the above formula:

$$\text{Offset} = \frac{(t + p1 + b) - (t + p1 + b + p2 + e) + (t + p1) - (t + e)}{2}$$

This reduces to

$$\text{Offset} = e + \frac{p2 - p1}{2}$$

Therefore, the calculated offset is an approximation of the actual offset. The error term in the estimated offset is $(p2 - p1)/2$. This is half of the difference in the propagation times of the message. Ideally, if the time taken by the message to go from the client to the server is the same as the time taken for the replied message to propagate from the server to the client, the calculated offset will be exactly equal to the actual offset. In the worst case, when $p1 >> p2$ or $p2 >> p1$, the error will be bounded by D/2 where D is the roundtrip delay given by

$$D = (T4 - T1) - (T3 - T2).$$

3. Mobility and dynamic topology of MANets

   Unlike the Internet, the locations of nodes in a mobile ad hoc network changes. This has to be considered in implementing an NTP based algorithm in MANets. In NTP, computers are organized into a hierarchy, which is somewhat preconfigured. Since a node remains stationary, with respect to its relative location with its neighbors, it is possible to define the hierarchy for clock synchronization where clock information is passed from primary and secondary sources to lower levels. Such a hierarchical structure that is predefined does not seem like it is suitable for MANets. Therefore, it may be worthwhile to consider an alternative approach to creating a hierarchical structure or selecting servers.

4. Partitions

   In a mobile ad hoc network, individual nodes or a group of nodes often separate from the rest of the network creating a partition. Partitions can occur in static wired networks due to failures. However, partitions are much more frequent in mobile ad hoc networks and proposed algorithms may have to consider it for better performance.

5. Other Issues

   There are several other issues that can affect the success of an NTP based algorithm for MANets. An example of such a concern is the situation when more than one server is available to a node for synchronization. It may also be interesting to explore possibilities of using a symmetric mode of synchronization sometimes over a client-server mode. In symmetric mode, nodes both request from and provide time information to their peers.

NTP Adapted Algorithm

We propose this algorithm as a potential solution to the problem of clock synchronization in mobile ad hoc networks. It incorporates some characteristics of the Network Time Protocol [9] and self-stabilizing spanning trees [17] while considering the issues discussed above. The Network Time Protocol (NTP) is the Internet standard for clock synchronization. Self-stabilizing spanning tree algorithms define a tree in a network that includes all the nodes and can recover from transient faults that may result from changes in network topology.

Description: A mobile ad hoc network is assumed to have at least one node that has access to a precise time source such as a GPS (Global Positioning System) receiver. Such a node with superior clock information is called a primary server. This algorithm assumes that a network has one primary node. All nodes possess a unique node ID, a list of neighbors and an integer field called *level*. A level is assigned to a node and indicates its proximity (minimum number of hops) to the primary server. Initially, the primary node is assigned a level of 1. The *level* field of all other nodes is set to the number of nodes in the network, n (or a number greater). This algorithm is based on the idea that each node in a MANet should obtain clock information from its 'best' neighbor. Here, 'best' is determined by proximity (minimum number of hops) to the primary server of the network. After every iteration, which occurs every m seconds (where m denotes a duration of time that can be changed), nodes select neighbors to synchronize with. To select a time source, a node searches through its list of one hop neighbors for the node with the lowest level. It compares this to its own level. If its level is higher in magnitude than the lowest level of a neighbor, the node sets that neighbor as its time source and sends it clock synchronization request. The server or time source returns the message to the client with clock information. It also stamps the message with its most recent level. The client synchronizes its clock, updates its neighbor information and sets its own *level* field to be one higher than the level of its server. Every node undergoes this synchronization process periodically. Thus, gradually and eventually, a hierarchy will be established among nodes connected directly or indirectly to the primary server. To address partitions, nodes or groups of nodes that have separated from their servers and are not connected to any other nodes that are connected to the primary server will undergo deterioration of their levels.

Deterioration and handling partitions: Deterioration of levels is a mechanism by which this algorithm seeks to handle changes in network topology. It allows the algorithm to dynamically modify the clock synchronization hierarchy needed due to movement of nodes. Since, smaller *levels* indicate greater proximity (smaller number of hops) to the primary node, deterioration corresponds with an increase in the level of a node to reflect a change in the nodes position with respect to the primary node.

By changing the way deterioration occurs, the algorithm can be made to handle mobility and partitions differently. One approach would be to let the clocks of partitioned nodes float or drift independently. Thus, the algorithm will not attempt to synchronize these nodes. This may seem reasonable, as certain applications may be able to ignore partitions. A simple implementation of this would increment the level of a node (except the primary server) by one every time it fails to find a neighbor to synchronize with.

Thus, $\text{level}_i = \text{level}_i + 1$.

Alternatively, it may be important to synchronize the clocks of portioned nodes with each other. This can be implemented by using the above scheme of incrementing the *level* field of nodes, and allowing nodes at the same level to exchange clock information. This synchronization of clocks among nodes at the same level may be done in a manner similar to NTP's symmetric mode. A possible way to implement this may be to synchronize a partitioned node's clock to the average of the clock times on its one-hop neighbors including itself. Depending on the application, more complex deterioration functions may be developed and used.

The pseudocode for a basic version of this algorithm is given below. .

Pseudocode:

```
    for(all n  nodes @ syncunit) { // syncunit is a unit of time for synchronization
    mode = clientserver; // default
        if (mylevel != 1){
        for (all 1hop neighbors)
                find lowest neighbor level;
        if (lowest neighbor level < mylevel){
        send synchronization message to node;
        }
        else{
        for (all 1hop neighbors: i)
                if (mylevel==level i)
                mode = peer; // symmetric mode
                send synchronization message to node;
        Mylevel = Mylevel + 1;
        }
}

        On receiving reply from server
        If( mode ==client server)
                Mylevel = serverlevel + 1; //
                adjust clock; [ADD message delay estimation]
        If (mode ==peer)
                Set time to avg of all 1hop neighbors (including self);// include
                                                              delay estimation
                My level = My level + 1;
```

To illustrate how the algorithm works, an example is discussed. Consider the network shown in figure 2 with six nodes, P_0 through P_5. During initialization, P_0 has a level of 0 while all the other nodes have their *level* field set to the number of nodes in the network, 6. In the first iteration (Figure 3), every node P_i searches through its neighbor list to find a neighbor with a level smaller than the P_i's own level. Nodes P_1, P_2 and P_4 find node P_0 to be a neighbor with a smaller level (equal to 0). They therefore send synchronization messages to P_0. Nodes P_3 and P_5 do not find any neighbor to have a lower level and therefore

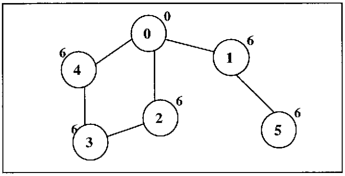

Figure 2: Example network initialization. The level of the primary node is 0. All other nodes are initialized to levels of 6. Lines between nodes indicate that they are within communication range of each other.

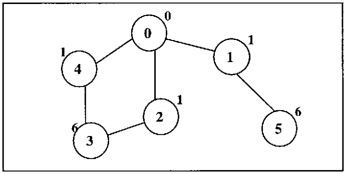

Figure 2: Example network after first iteration. The level of node 0 is 0. The level of node 4, node 2 and node 1 is 1. The level of node 3 and node 5 is 6.

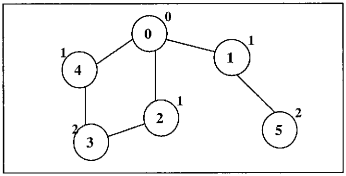

Figure 4: Example network after second iteration. Node 3 and node 5 have levels of 2.

increment their own levels. When nodes P_1, P_2 and P_4 receive replies from P_0 they adjust their clocks and set their own *level* fields to 1. In the next iteration, P_1, P_2 and P_4 send a synchronization message to P0 (Figure 4). This time, P3 finds P2 and P4 finds P1 to be neighbors with lower levels. P4 therefore sends a message to P1 for synchronization. However, P3 has two neighbors with lower levels. Since both P2 and P4's levels are the same, it picks the one with the lower id. Therefore, it sends a synchronization message to P2. When P3 and P5 receive replies, they both set their levels to 2. Considering mobility, let P1 move as shown in Fig 5. It no longer is in communication range with P0. In the next iteration, P5 with a level of 2 will find P1 with a level of 1 and send it a synchronization message. However P1 will only have P5 as a neighbor. Therefore, its level will undergo deterioration and increment by one. Over time, the levels of both P1 and P5 will keep increasing. If symmetric mode

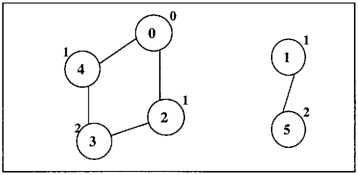

Figure 3: Mobility and Partitions. Mobility causes nodes 1 and 5 to be partitioned.

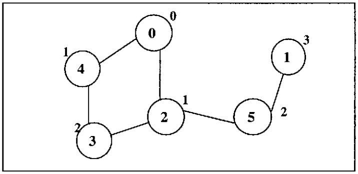

Figure 4: Final network topology.

partitions are handled using a s Suppose if P5 were to move and come into communication range with P2, the partition were to rejoin the rest of the network. Now, P5 will find P2 to be the node with the lowest level among its neighbors and itself and its neighbors and send it a synchronization message. After it receives a reply, it will set its level to 2. In the next iteration, P1's level will be 3. Thus the final network topology will be as shown in Fig 6.

As may be apparent from the discussion above, this algorithm can also be applied to networks with more than one primary node. In this case, all the primary nodes will have levels of zero and will be servers to the other nodes.

Potential strengths and weaknesses of algorithm: This is a simple algorithm that has a relatively low overhead and may be suitable for a variety of applications. It is based on NTP and is distributed. It is self-stabilizing as it can recover from changes in network topology that may result in invalid states. It appears to be promising for MANets, especially if the message delays are small. Unlike NTP, it does not require any pre-configuration (pre-defined hierarchy). Also, it does not require algorithms such as those for approximating Minimum Connected Dominating Sets such as the *broadcasting* algorithm, which is discussed later. Its self-stabilizing property makes it suitable for networks with dynamic topology changes, and the algorithm may be able handle partitions effectively.

A drawback of this algorithm is the potential problems it may have if message delays become significant and in the worst case, if delay estimation is not very tight or accurate.

Simulation and Results

Network Simulator 2 (ns-2) was used for simulation and testing. Ns-2 is a discrete event simulator that is widely used in network simulations. The NTP adapted algorithm was tested under various types of network conditions.

Model: A mobile ad hoc network with an area of 1000 x 1000 meters was simulated. Nodes were placed in the network, randomly. Every wireless node has a bandwidth of 2 Mbps and a radio transmission radius of 250 meters. Different scenarios with 50, 75,100,125 and 150 nodes were tested. Each of these scenarios was tested with different node mobility levels. These mobility patterns were based on the Random Way Point (RWP) model of ns-2. These ranged from a static case in which nodes did not move to higher mobility levels where nodes moved at average speeds of 5 m/s, 10 m/s and 15 m/s. Every node has an *offset* field that is initialized by a random value chosen between – 0.001 and 0.001 time units. The value of the logical clock at a node is determined by the sum of the clock time of the ns-2 simulator (assumed to be the real time) and the value in the *offset* field.

The NTP adapted algorithm is divided into two components: the spanning-tree component and the clock synchronization component. The spanning-tree component operates on the network periodically and makes or updates a spanning-tree. The clock synchronization component runs the NTP adapted clock synchronization algorithm. This algorithm is run periodically. Both algorithms are implemented in the agent layer of ns-2 and use DSR as the routing protocol.

Results: The simulation for each network scenario was run for 500 time units. The offset for each node was measured every 50 time units.

Figures 7, 8, 9, 10 and 11 show the average of the values of the offset measured at 70, 120, 170, 220, 270, 320, 370, 420 and 499 time units. Results are shown for different network sizes with varying levels of node mobility (average node speeds of 0m/s, 1m/s, 5m/s, 10m/s and 15m/s).

These results show that the average offset measured is somewhat high and displays a rather random pattern with occasional spikes.

Analysis of Results: The average of the offsets measured for all the nodes are in general, high. Thus, the results reveal that this algorithm (in its tested form) performs poorly in Mobile Ad Hoc Networks. This may be due to a number of factors, such as:

Variable Message delays: The most likely explanation for poor synchronization results is the highly variable nature of message delays in mobile ad hoc networks. As discussed earlier, the error in the offset estimated by the synchronization algorithm is equal to half the difference in the propagation times of a message from the client to the server and from the server to the client. In a system with high variability and large propagation times, this error can be significantly high and can reduce clock synchronization performance. In the worst case, if the error in the offset estimation is very high, it can increase the difference between the logical clock time at a node and the real time, thus degrading the synchronization of the clock at the node with respect to real time.

Outliers: In the present form, the algorithm lacks a means of discarding outliers or clock synchronization messages that cause huge offsets in the logical clocks at nodes. For example, in figure 10, the spike seen at 150 time units results from a node (Node ID 27) with an extremely high offset (27 time units).

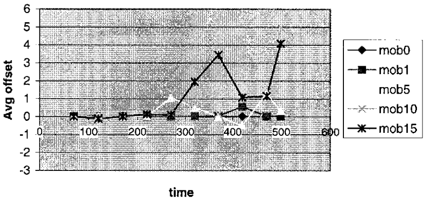

Figure 7: Average offset for 50 node network.

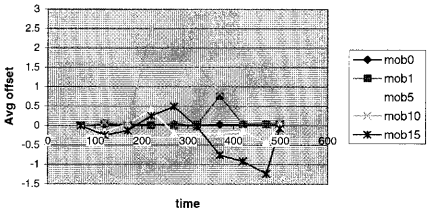

Figure 8: Average offset for 75 node network.

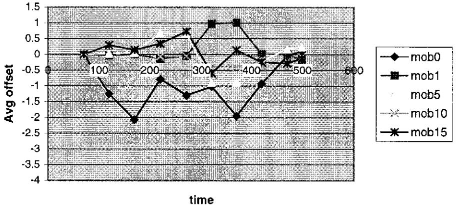

Figure 9: Average offset for 100 node network.

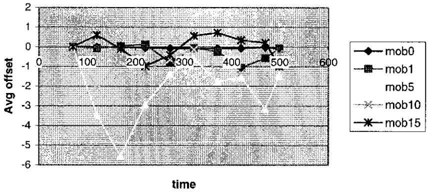

Figure 10: Average offset for 125 node network.

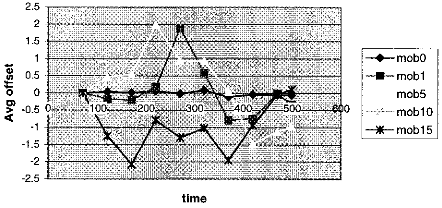

Figure 11: Average offset for 150 node network.

Improvements

As the simulation results reveal, the algorithm tested in its present form may not perform very well in mobile ad hoc networks because of reasons discussed earlier. However, modifications can be made to this algorithm that may improve its suitability to MANets.

One possible improvement involves the incorporation of a method of selecting messages that can be used for synchronization. To better illustrate this, consider two nodes, A and B, where B is a server for A. Since A will seek to synchronize its clock with B's, it will send synchronization messages periodically to B. B will timestamp these and return them to A. In its present form, the NTP adapted algorithm will use all message replies received by A to determine the estimated offset at A, for clock synchronization. Since the error in the estimated offset has an upper bound given by half the roundtrip delay of the message, messages with large roundtrip delays can have produce large errors in offset estimation. This may lead to poor synchronization clock of A's clock. However, if a threshold value for the roundtrip delay is determined, such that all messages above this value are discarded, the messages actually used for clock synchronization will have low roundtrip delays. This may reduce the error in offset estimation and improve the performance of the algorithm.

The simulated algorithm may also improve in performance if the bandwidth used by it is reduced. Presently, the implementation of the NTP adapted algorithm uses two components for the spanning tree and for clock synchronization. Each of these components uses its own set of messages. Thus, a high volume of messages may be sent between nodes, increasing the likelihood of collisions and transmission delays. This may in turn degrade the performance of the algorithm. Decreasing the number of messages sent, by combining clock synchronization messages with those used to determine the spanning tree or by piggybacking one on the other, can reduce the variability and magnitude of message delays and achieve better simulation results.

BROADCASTING AND MCDS ALGORITHMS

Another approach to clock synchronization in mobile ad hoc networks involves the use algorithms that can take advantage of certain characteristics of mobile ad hoc networks. Algorithms such as Reference Broadcast Synchronization (RBS) proposed by Elson et al, take advantage of the broadcast communication property of MANets [13, 18]. Briefly, all nodes receiving a broadcast message, receive it at almost the same time. Nodes exchange clock information using the time of receipt of the message by the nodes as a reference. This method obviates the need for estimating message delays, which may be difficult to measure in MANets. However, RBS, by itself, is limited to one broadcast domain [18]. To extend it to multiple hops, Minimum Connected Dominating Set (MCDS) approximation algorithms may be used. MCDS algorithms identify a subset of the network, such that, every node is in the subset or a neighbor of a node in the subset. Therefore, all nodes can receive broadcasts from nodes in the MCDS approximation subset, extending the RBS algorithm.

While there are several MCDS approximation algorithms [11, 14, 15, 16] it is interesting to consider the strengths and weaknesses of these algorithms and analyze how suitable they may be for MANets.

Four potential algorithms are described in Cheng and Du [14], Chen and Liestman [15], Dubhashi, et al [11] and Wu and Li [16]. Figure 7 compares these four algorithms using four criteria that were thought to be possibly important in mobile ad hoc networks.

Algorithm	*Performance*	*Bandwidth*	*Time*	*Inf*
Cheng and Du [14]	8 x (Optimum)	n	O (nΔ)	1 hop neighbors
Chen and Liestman [15]	log (Δ) x (Optimum)	-	-	1 hop neighbors
Dubhashi, et al [11]	log (Δ) x (Optimum)	-	O (log n)^2	Know graph size
Wu and Li [16]	-	O (nΔ)	O (Δ)^2	2 hop neighbors

Figure 12: Comparison of Algorithms. Here, n denotes the number of nodes in the network and Δ is the maximum node degree (number of edges). A hyphen indicates that no analysis is available in the paper.

In the table, *performance* refers to how good the algorithm is in approximating a minimum connected dominating set. The *bandwidth* refers to the amount of messages that the algorithm sends between nodes for MCDS approximation. The *time* field indicates the time complexity of an algorithm. The *inf* field lists if an algorithm makes any assumptions about information that nodes have about their neighbors or the network.

From the comparison table, the algorithm proposed by Cheng and Du can be expected to have a good performance. A higher performance suggests that the connected dominating set produced by the algorithm will be smaller. This is desirable, as, fewer nodes will broadcast to other nodes, which may result in fewer collisions. Also, the *bandwidth* and *inf* fields suggest the amount of resources the algorithm will require to run. Thus, in a mobile ad hoc network, if bandwidth is at a premium, it may be practical to tradeoff performance for saving bandwidth. Another important factor is the time complexity of the algorithm. For large networks with high mobility, faster algorithms may achieve significantly better clock synchronization when compared to slower algorithms.

Therefore, it maybe important to test different MCDS approximation algorithms in conjunction with the Reference Broadcast Synchronization algorithm to determine suitable MCDS algorithms for MANets.

CONCLUSION

The results obtained from this research reveal that the NTP adapted algorithm, in its present form, performs poorly in synchronizing clocks in MANets. Unlike the broadcasting algorithm, which utilizes the broadcasting property of mobile ad hoc networks, the NTP adapted algorithm relies on estimating message delays (and therefore an estimate of the difference between the clock time at a node and the real time), which may be highly variable in mobile ad hoc networks. Therefore, this research shows that the simulated NTP adapted algorithm, may not be very suitable for MANets as NTP is designed for wired networks.

However, more importantly, this work also suggests how the NTP adapted algorithm can be modified and improved to potentially perform better in mobile ad hoc networks.This, in addition with the simplicity and low overhead of the algorithm may make it a suitable alternative for clock synchronization in mobile ad hoc networks.

REFERENCES

1. N. Malpani, J. Welch and N. Vaidya, "Leader Election Algorithms for Mobile Ad Hoc Networks" Proc. International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M), pp. 96 – 103, August 2000.
2. J. E. Walter, J. L. Welch and N. Vaidya, "A Mutual Exclusion Algorithm for Ad Hoc Networks", 1998 Dial M for Mobility Workshop, October, 1998.
3. D. L. Mills, "Internet Time Synchronization: the Network Time Protocol", IEEE Transactions. Communications vol. COM-39, no. 10, pp.1482-1493, October 1991.
4. S. Keshav, "An Engineering Approach to Computer Networking", Addison Wesley Publishing Company. pp. 3-41, 1997.
5. N. Vaidya, "Tutorial on Mobile Ad Hoc Networks: Routing, MAC and Transport Issues", http://www.cs.tamu.edu/faculty/vaidya/presentations.html, July 10, 2001
6. U. Windl, et al, "The NTP FAQ and HOWTO -Understanding and using the Network Time Protocol (A first try on a non-technical Mini-HOWTO and FAQ on NTP)" http://www.eecis.udel.edu/~ntp/ntpfaq/NTP-a-faq.htm, July 20, 2001.
7. G. Shipley, "Getting in Sync: A Look at NTP" http://www.networkcomputing.com/1002/1002ws1.html, March 20, 2003
8. T. Srikanth and S. Toueg, "Optimal clock synchronization", Journal of the Association of Computing Machinery, Vol. 34, No. 3, July 1987, pp 626-645.

9. J. Lundelius and N. Lynch, "An Upper and Lower Bound for Clock Synchronization," Information and Control, Vol. 62, Nos. 2/3, pp. 190-204, 1984

10. Institute for Telecommunication Sciences, "Telecom Glossary 2K", http://www.its.bldrdoc.gov/projects/devglossary/_gps.html, March 22, 2003.

11. D. Dubhashi, et al, "Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons", ACM-SIAM Symposium on Discrete Algorithms, 12-14 Jan, 2003.

12. K. Romer, "Time Synchronization in Ad hoc Networks", ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp 173-182, October 2001.

13. J. Elson, L. Girod and D. Estrin, "Fine-grained network time synchronization using reference broadcasts", Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI), pp 147-164, 2002.

14. X. Cheng and D.-Z. Du, "Virtual backbone-based routing in ad hoc wireless networks", Technical Report 02-002, University of Minnesota, Department of Computer Science and Engineering, 2002.

15. Y. Chen and A. Liestman, "Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks", ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), June 2002.

16. J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks", Proceedings of the 3rd international workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp 7-14, 1999.

17. G. Antonoiu and P.K. Srimani, "Distributed Self Stabilizing algorithm for minimum spanning tree construction", Technical Report, Colorado State University, May 1997.

18. G. Cao and J. Welch, "A Clock Synchronization Service in Mobile Ad Hoc Networks", Submitted for publication, 2003.

VITA

Rajan Chandra was born in Sydney, Australia in 1981. He later moved with his parents to India. After graduating as the valedictorian of his high school in India and among the top high school students in the country, he enrolled at Texas A&M University for undergraduate education. At Texas A&M University, he has been received several honors and awards including the Joe L. Cooper scholarship (2000-2002), the Robert M. Kennedy scholarship (2002), the J. Van Dyke scholarship (2002), the Texas Telecommunications Engineering Consortium Undergraduate scholarship (2001) and the Dean's Honor Award (2000, 2001). He is currently a senior majoring in Computer Engineering in the Department of Electrical Engineering at Texas A&M University.

Permanent Address:
5110 Pine Street
Bellaire TX 77401
USA