Time Synchronization in Sparse and Highly Mobile Sensor Networks

By Chompoonoot Veerakiatikit

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Xavier Défago

September, 2011

Time Synchronization in Sparse and Highly Mobile Sensor Networks

By Chompoonoot Veerakiatikit (0910203)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Xavier Défago

and approved by
Associate Professor Xavier Défago
Professor Mizuhito Ogawa
Associate Professor Azman Osman Lim

August, 2011 (Submitted)

Abstract

In mobile sensor networks, many applications depend on the availability of a global time reference. For instance, the order of occurrence of events detected by different sensors may affect the interpretation of the data. This can be done easily with timestamps, but requires that clocks be properly synchronized. The system with lack of appropriate synchronization will operate in wrong condition, or even ends up with causing failure of the system. To solve this problem, time synchronization protocols for sensor networks is used in order to maintain clock synchronization in the system. Since existing protocol cannot provide the efficient solution for mobile system, this research presents two different scheme protocols for sparse and highly mobile sensor network. First, Mobility Prediction Time Protocol (MPTP), which utilizes the method of mobility prediction to estimate the connection lifetime between any two nodes, then follows the proposed parent choosing criteria in order to construct the strong connected time synchronized hierarchical topology in the system. MPTP tries to make the rare change in the topology and provides dynamic connection re-establish mechanism before any connection loss. Second, Population-based Time Protocol (POP-B), which is adapted from opportunistic scheme protocol to spread the clock information over the entire network. POP-B utilizes the high opportunity to meet other nodes due to node's high mobility, thus, the reference clock information can be quickly spreaded and cover all though the system.

To measure the performance of protocol, we derive analytical model and conducted the simulations. SNTP, the basic hierarchical time synchronization protocol in sensor network, and RTSP, the protocol which maintains node list to tackle the mobility problem, are simulated as the reference protocol to compare against to. The simulation results show that MPTP and POP-B can achieve very high clock synchronization accuracy and stability compared to SNTP and RTSP, and both protocols can perform even better when mobility increases. However, POP-B, which uses opportunistic and non-structural scheme provides the higher accuracy, and even more stable than MPTP does.

Keywords: Time synchronization; High mobile sensor networks; Sparse network; Mobility prediction; Opportunistic protocol.

Acknowledgements

First and foremost, I wish to express my most sincere gratitude to my supervisor, Associate Professor Xavier Défago, School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), for his kindly guidance, constant encouragement, and helpful support. He is my inspiration. Many meaningful quotes I got from him, Such as, "The most important thing is to love, enjoy and have fun in what you do" and "The perfect doesn't really exist. You can do even better, better and better to improve yourself". I am really lucky and so proud to be one of his students.

I would like to express my thank to Associate Professor Azman Osman Lim, the advisor of my sub-theme research, for his helpful suggestions and comments on my sub-theme research. He introduced me the mobility prediction idea on network routing protocol, which inspires me a new idea to solve my problem in my main research later. Without his useful guidance about research and international student life experiences, my two years of living in Japan would be the hard time in my life.

I would like to thank Professor Mizuhito Ogawa for his advices and for taking time from his busy schedules to serve on my dissertation committee.

My special thank to Career Development Program for Foreign Students in Japan of JAIST, who provided me financial support and Japanese language study for life in Japan.

I would like to thank my seniors and friends in Défago-Lab, Daiki Higashihara, Bonnet François and Shintaro Hosoai for their kind helps, discussions and suggestions. Thank you for all supports and warm friendship during the whole period of studying in Japan.

Along the way, innumerable people either directly or indirectly have provided me knowledge, experience and support, I would like to thank them all.

Last, but not least, thank you my family for being my motivation, love and support. Thank you Waris, M, Heng, and Bier for walking together beside me, and raise me up whenever I am down over two and a half years of being away from home. Thank you Chris, Amara, Nee, and Oat for advices and inspiration. Thank you all of my Thai friends in JAIST who support and encourage me all the time. Also, thank you all of my friends and teachers in Thailand.

Without their continuous supports, my research would not have been smoothly completed. This research is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in miniaturization design have led to active research in small, wireless and low-power highly distributed sensor networks. Sensor nodes typically consist of three important components; sensing component, data processing component, and communicating component, which allow the new concept of creating smart system based on collaborative effort of a large number of nodes. For example, smart sensor node can be embedded in appliances, such as vacuum cleaners, microwave ovens, and refrigerators. These sensor nodes can interact with each other to provide services, and also allow remote system management.

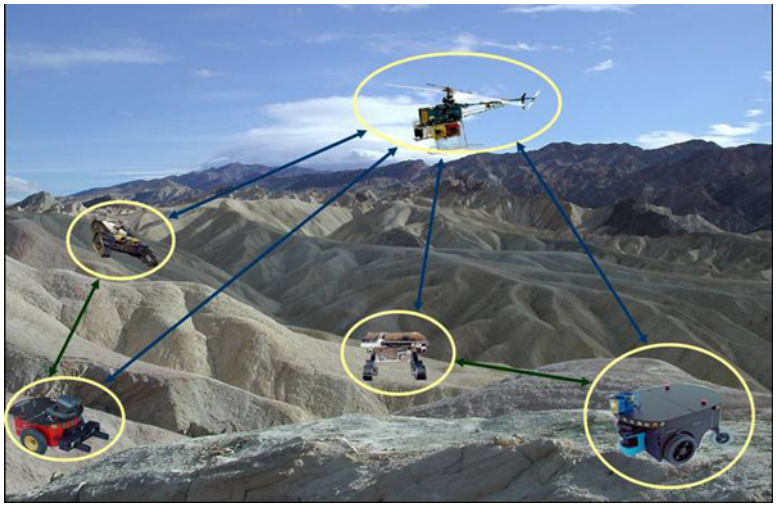


Figure 1.1: The investigating system

In mobile sensor networks, many applications depend on the availability of a global time reference; for instance, the order of occurrence of events detected by different mobile sensors may affect the interpretation of data, such as, investigating system (shown in Figure 1.1 [MAR]), which deploys a group of robots to visit checkpoints in the area. This kind of application can be done easily by recording timestamp, but requires that clocks must be properly synchronized, otherwise data will be mistranslated.

The system with lack of appropriate synchronization will operate in wrong condition, or even end up with causing failure of the system. To solve this problem, time synchronization protocols for sensor networks are used in order to maintain clock synchronization in the system. However, few existing protocols have mentioned about time synchronization in mobility model assumption. Since reference node availability drops due to frequent changes in topology caused by mobility, existing time synchronization protocols still cannot provide the efficient solution for mobile system.

1.1 Objective

Our research concerns about finding time synchronization protocol for **sparse** and **highly mobile** sensor networks. Since low density network and mobility cause frequent changes to network topology, preserving synchronized system becomes more difficult as time reference node is rarely available. Our proposed protocols try to tolerate topology changes, so that the system clock can maintain in stable state, and can recover itself even when some critical event (such as node reset) occurs.

1.2 Contributions

There are two contributions in our thesis. The first one is the design of mobility prediction method assisted time synchronization protocol in sparse and highly mobile sensor networks. The second contribution is the new approach of using opportunistic concept in time synchronization protocol in sparse and highly mobile sensor networks.

Mobility Prediction Method Assisted Time Synchronization Protocol

Mobility prediction method was proposed in concept that mobile node's future location and network topology changes can be predicted accurately enough, so that route reconstruction can be done prior before changes occur. By using the concept of prediction, we propose hierarchical time synchronization protocol, which constructs the strong connection among nodes inside topology by using mobility prediction to select strong reference node.

Opportunistic Concept in Time Synchronization Protocol

The proposed mobility prediction method protocol can achieve good performance up to certain level. However, under the extremely change environment, the protocol faces a diffucult problem in maintaining synchronization topology. As a result, instead of

emphasis on improving node availability and increasing strength of topology to handle topology changes, we change our viewpoint by looking time synchronization as the flow of data in the system. The proposed protocol is based on a population protocol model. This scheme also works very well under high mobility system.

Evaluations

To measure the performance of protocol, we have derived analytical model and conducted the simulations. SNTP, the basic hierarchical time synchronization protocol in sensor network, and RTSP, the protocol which maintains node list to tackle the mobility problem, are simulated as the reference protocol to compare against to. The simulation results show that both protocols can achieve very high clock synchronization accuracy and stability compared to SNTP and RTSP, and can perform even better when mobility increases. However, the opportunistic and non-structural scheme protocol provides the higher accuracy, and even more stable than mobility prediction method protocol does. Mobility prediction method protocol can perform at 100 and 200 times better clock accuracy than SNTP and RTSP, while opportunistic protocol can perform at 250 and 500 times better clock accuracy than SNTP and RTSP respectively.

1.3 Structure of the Report

The rest of the thesis is organized as follows: In Chapter 2, we introduce sensor networks characteristics and details of recent time synchronization protocols. Chapter 3 depicts the problems of time synchronization protocol in mobile sensor networks. The system model and definitions are presented in Chapter 4. Chapters 5 and 6 describe the two proposed protocol; Mobility prediction assisted time synchronization protocol and opportunistic time synchronization protocol. The simulation model and results are shown and discussed in Chapter 7, while Chapter 8 concludes the research.

Chapter 2

Related Work

2.1 Sensor Networks

Sensor networks consist of distributed autonomous sensors to monitor physical or environment condition. The origin of sensor networks development was first motivated by military applications. Until now, sensor networks are used in many industries or even in home appliance system.

Sensor node typically consists of three components [IFWYE02]:

1. Sensing component
   Sensor node equips the special electronic part for monitoring and measuring target's attribute.

2. Data processing component
   The computation unit is used for processing data in order to exchange among nodes in the system.

3. Communicating component
   A radio transceiver allows interaction with other nodes in the system for triggering or exchanging data.

During operation, sensor node uses sensing component for data measuring such as temperature, motion, and brightness. Then, by using data processing unit, sensor node prepares the raw data into the decided format in order to exchange with the other nodes. Radio transceiver in communication component is used for connection establishment, sending data among nodes in system.

2.1.1 Characteristics

Power consumption is one of the most important constraints in sensor node. Because sensor nodes carry limited and irreplaceable power sources, protocols for sensor networks must focus primarily on power conservation. The trade-off between high Quality of Service

(QoS) and mechanism that give the end user the option of prolonging network lifetime must be concerned.

Typically, network infrastructure is unavailable in the deployed area, therefore, sensor nodes must have an ability to construct and manage network by themselves (**Ad hoc deployment**). Since crash and mobility of sensor node can cause topology change, recovery mechanism is necessary. Moreover, by **heterogeneity** of nodes, in the system, every sensor node has the same abilities and operates to the same task.

2.1.2 Sensor Networks Application

Sensor networks represent a significant improvement over traditional sensors, which are deployed in the following two ways:

1. Sensor can be positioned far from the actual phenomenon. In this approach, large sensors that use some complex techniques to distinguish the targets from environmental noise are required.

2. Several sensors that perform only sensing can be deployed. The position of sensors and communications topology is carefully engineered. They transmit time series of sensed phenomenon to the central nodes where computations are performed, and data are fused. Figure 2.1 shows the sensor network application architecture. Captured data at node A will be transmitted through the network until reach at gateway sensor network G. G, as the gateway between observation area and user area, aggregates received data and submits to user U.

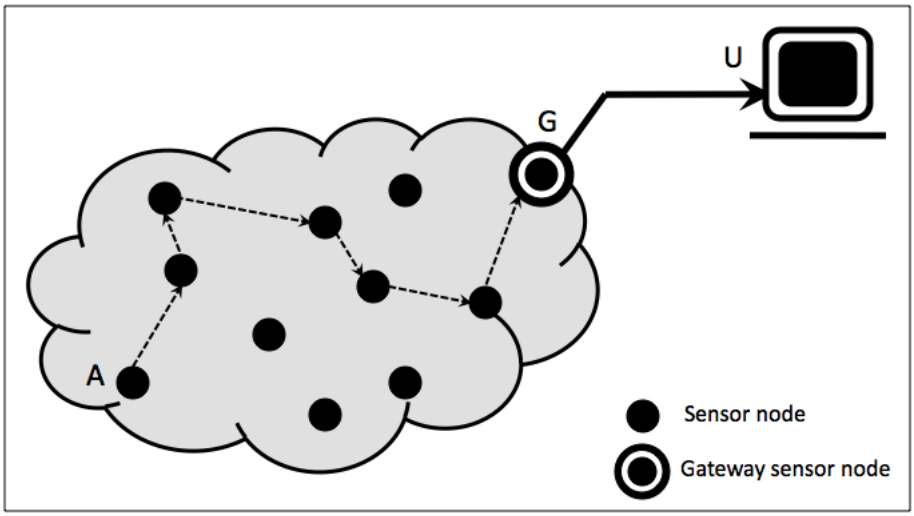


Figure 2.1: Sensor networks application architecture: Captured data at A is transmitted to gateway sensor node G, where data is aggregated and sent to user U's terminal

In addition, there are many different types of sensors, which are able to monitor a wide variety of ambient condition, such as, temperature, vehicular movement and lighting condition. Therefore, by using the concept of micro-sensing and wireless communication, this introduces us a number of applications, which can be categorized into eight main areas: military, environment, health, home, commercial, space exploration, chemical processing, and disaster relief.

2.2 Time Synchronization

Time synchronization is one of the most necessary topics in distributed system. Many applications in distributed system depend on the availability of global time reference. For instance, timestamp message will be transmitted among nodes in order to determine their relative proximity to one another. Especially, having common timing between nodes will allow for the determination of speed of a moving node. Unfortunately, hardware clocks are not perfect. The variations in oscillators cause the clock's offset; thus, the durations of time intervals of events are not observed the same between nodes. As the result, system, which lacks of appropriate synchronization, operates in wrong condition, or even end up with causing failure of the system.

Typically, time synchronization can be classified into two groups: Time synchronization in infrastructure networks, and time synchronization in Ad hoc networks (including sensor networks)

2.2.1 Time Synchronization in Infrastructure Networks

For infrastructure networks, there are two methods of time synchronization which are the most common [MROCHE06]:

- Network-based time synchronization protocol
  This kind of protocol enables accurate time stamping of data packets transferred between applications on the infrastructure network. For example, Network Time Protocol (NTP).

  The process of clock synchronization in NTP [NTP3] is done with client-server fashion. Usually, in the system consists of one server with an atomic clock, and the synchronization is done by constructing the system hierarchy. We define computer node in each stratum (level), the upper level computer acts as server, and the lower level computer acts as client in synchronization process. When client computer wants to synchronize with server, it sends UDP packet requesting the time information. The server will then return timing information and, thus, the client will be synchronized.

- Signal-based time synchronization protocol
  Signal-based time synchronization protocol is a method for synchronizing clock by a time code bit stream transmitted by a radio transmitter connected to a time

standard such as an atomic clock. For example, Global Positioning System (GPS) and radio clock.

GPS [HR00] requires device to communicate with satellites in order to synchronize. The time accuracy of GPS depends on how many satellites the receiver can communicate with at a given time. This will not always be the same, so the time accuracy will vary. Moreover, GPS devices depend on line of sight communication to the satellite, which may not always be available where wireless networks are deployed.

Although there are many protocols proposed for infrastructure networks, existing protocols cannot be readily applied to Ad hoc networks due to limited resources, power and complexity.

2.2.2 Time Synchronization in Sensor Networks

The idea of designing time synchronization method in sensor networks is limited by sensor nodes constraints. While time synchronization in infrastructure networks aims for the best result in accuracy of clocks in the system, precise clock synchronization is not always essential in sensor networks. The definition of time synchronization does not necessarily mean that all clocks are perfectly matched across the network. Therefore, the purpose of designing of time synchronization protocol for sensor networks can be vary and is opened to meet one's needs.

Generally, time synchronization protocols for sensor networks can be divided into three main categories: flooding protocols, gossip-based protocols, and hierarchical protocols.

- Flooding Protocols
  This kind of protocol utilizes periodic flooding of synchronization message. The network structure is mesh type topology. Flooding protocol provides the ability for dynamic topology changes, and also allows robustness for node and link failure. However, this kind of protocol is not practically used as result of causing high-traffic in network. Figure 2.2 depicts the flooding method.

  E.g. Flooding Time Synchronization Protocol (FTSP) [MKSL04]

- Gossip-based Protocols
  This kind of protocol come from the analogy of office workers spreading rumors, which gets an advantage from flooding protocol since it reduces the network traffic. Figure 2.3 depicts the gossip-based method.

  E.g. Reference Broadcast Synchronization Protocol (RBS) [EE02]

- Hierarchical Protocols
  This kind of protocol uses a tree to organize the network topology. Protocol is conceptually broken up into two phases, the level discovery phase and the synchronization phase. The level discovery phase is done in order to create the hierarchical topology in which each node is assigned a level. Then, at synchronization phase,

the nodes with level i will synchronize with its parent nodes in level $i-1$. The hierarchical synchronization method is depicted in Figure 2.4.

E.g. Simple Network Time Protocol (SNTP) [SNTP4], Timing-sync Protocol for Sensor Networks (TPSN) [GKS03]

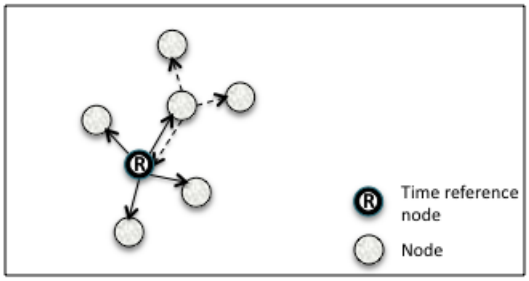


Figure 2.2: Flooding protocols

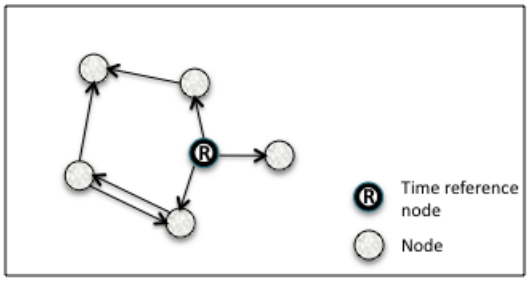


Figure 2.3: Gossip-based protocols

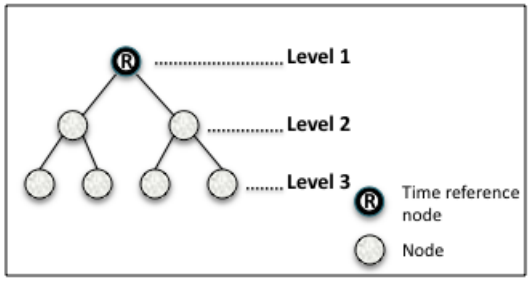


Figure 2.4: Hierarchical protocols

Simple Network Time Protocol (SNTP)

SNTP is a subset of NTP, which can be used when the ultimate performance of a full NTP is neither needed nor justified. SNTP uses paradigm for servers and clients. Servers are stateless and can support large numbers of clients, however, unlike most NTP clients; SNTP clients normally operate with only a single server at a time.

SNTP constructs hierarchical system of level of clock sources. Each level of this hierarchy is termed a stratum, and is assigned a layer number starting with 1 as primary server. The stratum defines its distance from reference clock, and exists to prevent cyclical dependencies in hierarchy. However, stratum is not an indication of quality or reliability at all. Figure 2.5 presents SNTP's structure.

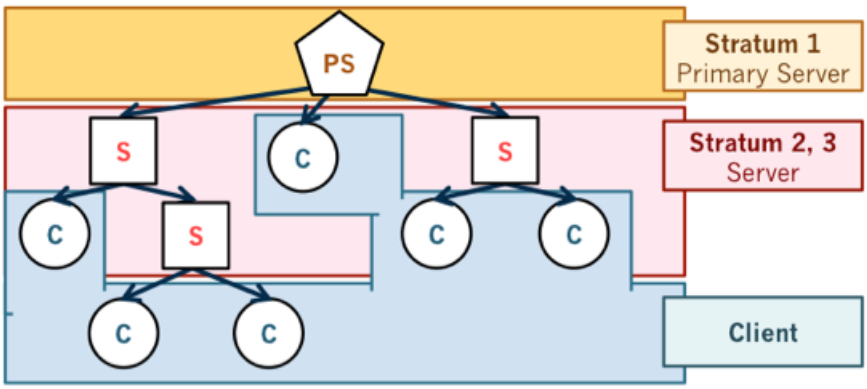


Figure 2.5: SNTP's structure

The mechanism of synchronization is showed in Figure 2.6. With 2-way time synchronization, client sends message with its timestamp $T1$ to server. At the time of receiving, server records timestamp $T2$ and replies message back to client with $T1$, $T2$, and adds timestamp a time message being sent, $T3$. When client receives reply message, it records timestamp $T4$, and then the synchronization procedure starts by calculating $delay(d)$ and $offset(t)$ from equations:

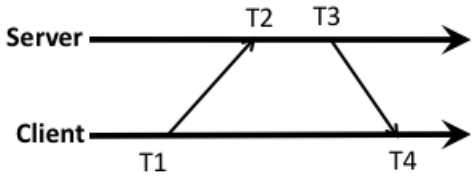


Figure 2.6: 2-way time synchronization

$$d = (T4 - T1) - (T3 - T2) \tag{2.1}$$

$$t = \frac{(T2 - T1) + (T3 - T4)}{2} \tag{2.2}$$

2.2.3 Time Synchronization in Mobile Sensor Networks

Although many proposed protocols aim at being used in sensor networks, all of these protocols were introduced without node mobility assumption. However, considering in

mobile sensor networks, movement causes topology changes frequently and may result in severe drop in clock accuracy, rising overhead, or even causing global instabilities.

Reliable Time Synchronization Protocol (RTSP)

To solve mobility problem in mobile sensor networks time synchronization, one attempt is Reliable Time Synchronization Protocol (RTSP) [HB05], which maintains parent candidate list in each node. When connection between node and its parent lost, instead of broadcasting message seeking for new parent, node immediately selects another parent in the candidate list and continues performing time synchronization.

RTSP assumes that nodes in the network have unique ID. As in NTP, in order to adjust clock, protocol calculates the roundtrip delay and clock offset between two nodes via exchanged messages timestamp. RTSP basically consists of two phases:

1. Hierarchical topology setup phase
   The hierarchical topology is created in the network at the first phase. The protocol constructs a tree with lower depth and generates candidate parent list, which is used to manage failure of nodes in the network.

2. Synchronization and handling topology change phase
   In the second phase, a node belonging to level i synchronizes with its parent node, which is belonging to level $i-1$ by exchanging timestamp messages. When a node cannot communicate with its parent, it selects another parent in the candidate list and performs synchronization.

Figure 2.7 explains how RTSP maintain the topology. Figure 2.7 (a) shows that every node in the system is holding its own candidate list. In the picture, node 6 holds node 0, 7, 11, 21's information in candidate list, and sorts priority according to their level in hierarchy respectively.

Figure 2.7 (b) shows the situation when the connection with node 0 (parent node) lost. Node 6 selects next candidate, node 11, and immediately constructs the connection with node 11.

Figure 2.7 (c), when node 6 can detect connection from new node (node 3), it will keep track the new nodes information in its candidate list.

RTSP mentions that reliability is improved. Synchronization error also decreases by creating lower depth tree. However, because information in candidate list is not updated real-time enough, RTSP can tolerate only rare topology changes.

2.2.4 Characterizing Time Synchronization

According to [JD01], important metrics, which used in studying time synchronization in sensor networks, are:

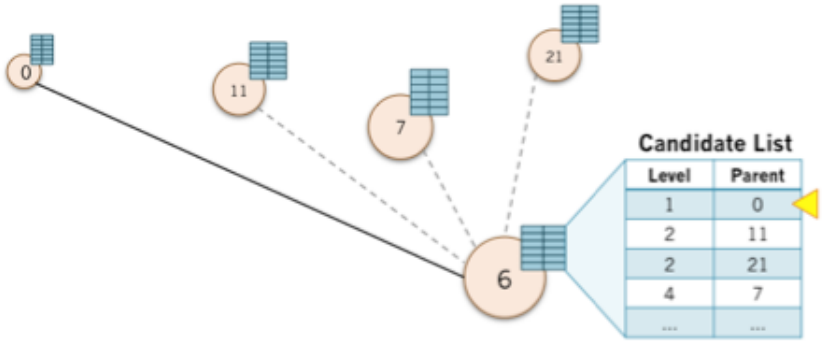
- Precision
  Either the dispersion among a group of peers, or maximum error with respect to an external standard

- Lifetime
  Lifetime, which can range from persistent synchronization that lasts as long as the networks operate to nearly instantaneous useful, for example, if nodes want to compare the detection time of a single event

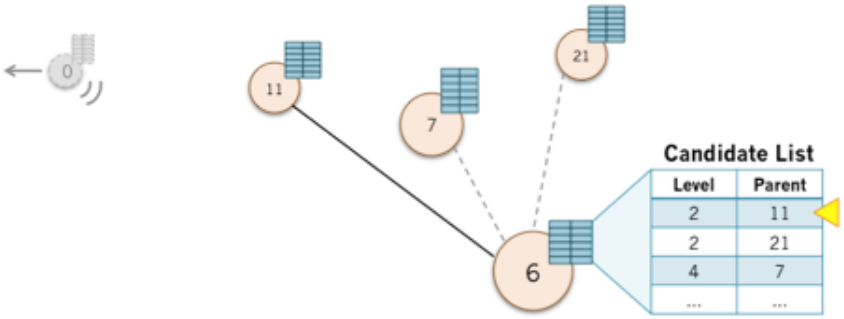
- Scope and Availability
  The geographic span of nodes those are synchronized and completeness of coverage within that region

- Efficiency
  The time and energy expenditures needed to achieve synchronization

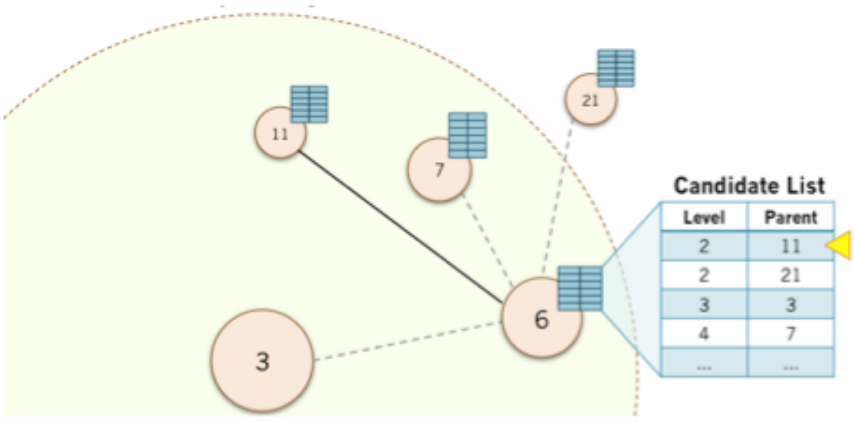
- Cost and Form Factor
  Cost and form factor, which can become particularly important in wireless sensor networks that involve thousands of tiny, disposable sensor nodes



(a) Each node maintains its own candidate list



(b) When connection with parent lost, node immediately uses the next candidate from the list



(c) When new node can be detected, node puts the new information into candidate list

Figure 2.7: RTSP's mechanism

Chapter 3

Models and Definitions

3.1 Communication Model

The system consists of a set of n nodes $\pi = p_1, p_2, p_3, ...p_n$. Every node has a unique ID. By using wireless communication, the set of neighbors for p_i is defined as $\eta = \{p_j | p_j$ is staying within p_i's transmission range $r\}$. Node can send and receive message to/from neighbor in η. Communication network itself cannot generate, duplicate or change the message. Message are not lost, and will arrive in FIFO order.

In the system, node p_i broadcasts message to send information. The necessary methods for sending and receiving message are:

$broadcast_{p_i}(m)$: Node p_i broadcasts message m to communication network.
$deliver_{p_j}(m, p_i)$: Node p_j receives message m sent from node p_i in communication netowrk, and delivers m to upper level.

Figure 3.1 shows the broadcasting process.

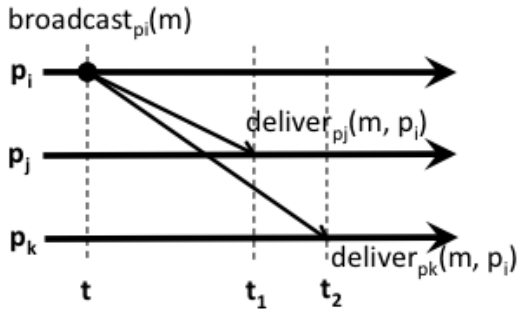


Figure 3.1: Broadcasting process

Every node p_i broadcasts message in every t_{hb} time interval. If node p_j does not deliver message from node p_i within t_{hb}, node p_i is suspended. Figure 3.2 depicts the message timeout situation.

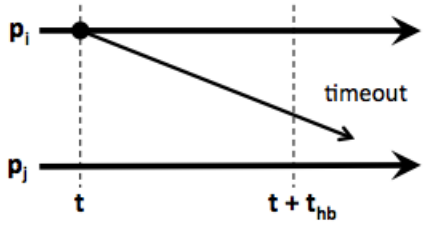


Figure 3.2: Message timeout

3.2 Clock Model

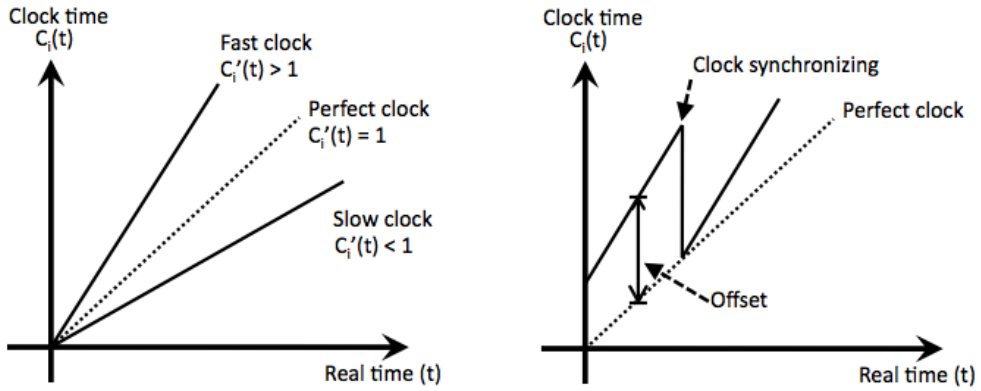
Each node maintains a clock as a function of hardware oscillator. The clock value of node p_i at time t is given by

$$C_i(t) = (1 + \phi_i)t + C_i(t_0) \tag{3.1}$$

where ϕ_i is a constant proportional coefficient of the node p_i oscillator (**clock drift**), and $C_i(t_0)$ is the initial clock value at time t_0.

In Figure 3.3 (a), the perfect clock relative to Coordinated Universal Time (UTC) is $\frac{dC_i(t)}{dt} = 1$. However, the clock deviates from perfect clock over time due to different in clock speed are $\frac{dC_i(t)}{dt} > 1$ for a fast clock, and $\frac{dC_i(t)}{dt} < 1$ for a slow clock.

Figure 3.3 (b) defines the difference in clock value $C_j(t) - C_i(t)$ as **clock offset**. To achieve high clock accuracy, **clock synchronization protocol** helps node adjusts its clock to minimize the clock offset among nodes in the network.



(a) (b)

Figure 3.3: Clock model: (a) Clock error caused by clocks running at different rates; (b) Clock offset and adjustment after synchronization

Chapter 4

Clock Synchronization Problem

In our research, we focus on clock synchronization problems in sparse and highly mobile sensor networks. Problem can be distinguished into two points of view: clock synchronization between neighbors, and entire system's clock synchronization.

Clock Synchronization between Neighbors

Due to the effect of node's high mobility, structured time synchronization protocol is facing with a number of connection loss between node and its reference. Hence, the performance of protocol drops from the rare availability of reference node. The reason can be divided into two sources:

1. Maintaining out-of-date reference node's information
   Since the topology in high mobility network changes frequently, by maintaining the static reference node's information method becomes the weak point to the system. The problem causes the miss query, and even causes the node absent from synchonization in amout of time.

2. Inefficient synchronization mechanism
   Typically, hierarchical time synchronization protocol consists of two phases (shown in Figure 4.1): topology construction phase, and time synchronization phase. However, due to the frequent topology changes, the two-phase scheme protocol cannot handle the time synchronization mechanism in time before the connection between nodes lost. As a result, we can obviously see that the protocol can be optimized by combining these two phases together.

Entire System's Clock Synchronization

Sparse network reduces nodes' opportunity to meet and synchronize clock. With a little chance to do clock synchronization, node must decide which information should be relied on in order to make the clocks in the system converge to the same value.

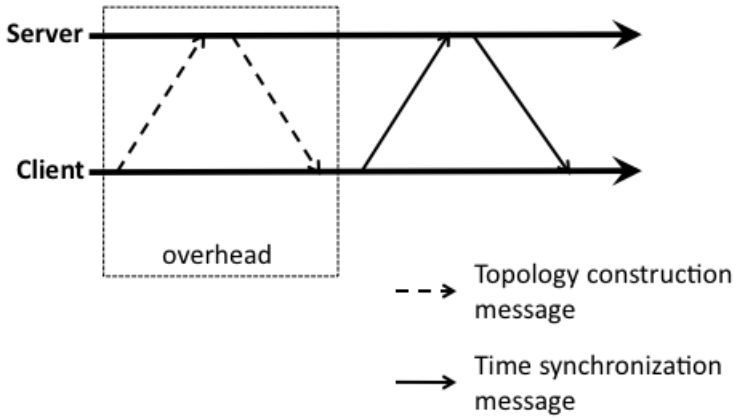


Figure 4.1: Overhead process: The topology construction mechanism cannot handle time synchronization process before connection loss

Chapter 5

Mobility Prediction Time Protocol

Mobility becomes a significant issue in clock synchronization problem. As mobility affects the performance of clock synchronization protocol by causing the frequent changes in network topology, clock reference node is rarely available for synchronization and, then, results in low clock accuracy in the system.

In this chapter, we propose Mobility Prediction Time Protocol (MPTP), which is a time synchronization protocol that utilizes the concept of dynamic and strong-connected topology construction to counter the mobility problem. The proposed MPTP is designed by constructing and maintaining the reliable connection between nodes inside topology. Mobility Prediction is used as approach as it utilizes node position to estimate the connection lifetime between any pair nodes.

MPTP is hierarchical time synchronization protocol. MPTP needs to construct topology and maintains the hierarchical synchronization; node at level $i + 1$ will synchronize with node in level i. Therefore, to achieve the better performance of time synchronization, choosing an appropriate reference and parent node should be necessarily concerned.

The design of MPTP considers the two following ideas:

1. Selecting the strongest reference node
   To handle node's mobility, MPTP enhances the reliability of reference node and synchronization process by selecting strong reference node. This is done by predicting reference candidate nodes' available time and then selecting reference node based on research proposed criteria. Starting with an arbitrary state, the connected nodes in the system will eventually construct a spanning tree topology with one decided root.

2. Broadcasting time synchronization message
   Based on SNTP's 2-way time synchronization method, MPTP broadcasts time message to number of nodes in every interval for clock synchronizing and updating network topology.

In our research, we term node's level in hierarchy as *stratum* like SNTP. Node at stratum 1 is defined as *root*, and acts as reference node by providing clock to system.

Nodes, which stay in others stratum but also provide clock for synchronization, are defined as *parent* node.

5.1 Mobility Prediction

Mobility prediction is the method of predicting the future state of network topology in order to perform route reconstruction proactively to minimize topology disruptions caused by mobility.

Typically, mobility prediction method can be categorized into two groups: mobility prediction method for fixed infrastructure type network, and for ad hoc network. However, prediction approaches for fixed infrastructure type are usually inappropriate in the case of ad hoc network from the following reasons [DCG10]:

1. Mobility prediction in fixed wireless networks is based on the use of a static underlying network infrastructure, while in ad hoc networks mobility prediction must be done in a highly dynamic environment, where the network topology is changing and the mobility of other nodes should be taken into consideration.

2. Ad hoc networks are usually applied in emergency operations and military environments, where future node movements cannot be based on a record of previous movements because of the dissimilar requirements of each situation.

3. Since mobility prediction methods for ad hoc networks are executed on the mobile nodes, they should be more lightweight than the methods for fixed wireless networks, typically executed on the base station.

Considering in ad hoc networks, [DCG10] classified mobility prediction methods into three categories: Movement history based prediction methods, physical topology based mobility prediction methods, and logical topology based mobility prediction methods.

5.1.1 Link Expiration Time (LET)

In our protocol, we implement Link Expiration Time (LET) estimation from physical topology based mobility prediction methods. According to [WSM01], LET method utilizes the location and mobility information provided by GPS. Since motion parameters of two nodes (e.g., speed, direction, radio propagation range, and position) are known, we can estimate the duration of time these two nodes will remain connected. Assume that two nodes i and j are within the transmission range r of each other. Let (x_i, y_i) be the coordinate of mobile node i and (x_j, y_j) be that of mobile node j. Also let v_i and v_j be the speeds, and θ_i and $\theta_j (0 \leq \theta_i, \theta_j < 2\pi)$ be the moving direction of nodes i and j, respectively. Then, the amount of time that two mobile nodes will stay connected, LET is predicted by:

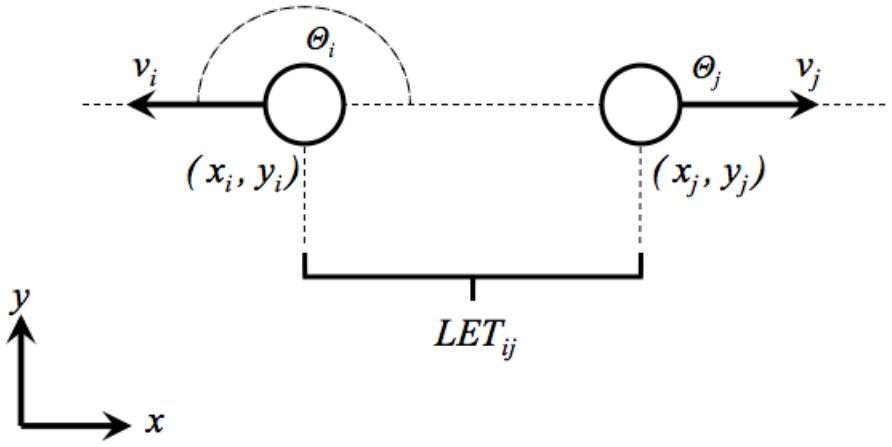


Figure 5.1: Calculating LET between nodes i and j.

$$LET_{ij} = \frac{-(ab+cd) + \sqrt{(a^2+c^2)r^2 - (ad-bc)^2}}{a^2+c^2} \tag{5.1}$$

where
$a = v_i \cos\theta_i - v_j \cos\theta_j$,
$b = x_i - x_j$,
$c = v_i \sin\theta_i - v_j \sin\theta_j$, and
$d = y_i - y_j$

Note that when $v_i = v_j$ and $\theta_i = \theta_j$, LET becomes ∞.

5.1.2 Strong Neighbor

In this research, we term *strong neighbor* as a neighbor node that is staying within the specified LET's bound. By using LET method, protocol tries to screen the neighbor, as parent candidate node, that will be available long enough to perform time synchronization, and can set up less frequently change topology when considering over entire system. The difference between strong neighbor and simple neighbor is decribed in Figure 5.2.

5.1.3 Method for Selecting Parent

To construct the strong connection topology, we proposed method for selecting node's parent based on the number of root's strong neighbor, the number of parent's strong neighbor, stratum and node ID respectively. Figure 5.3 presents the definitions used in MPTP. In the Figure, the network has A as a root, which is connecting with B, and has B as a child. Again, B is connecting with C, and has C as a child. In C's viewpoint, A

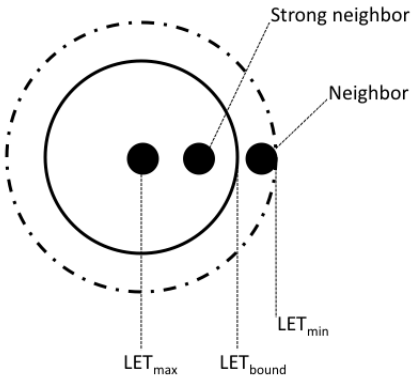


Figure 5.2: Strong neighbor and simple neighbor

is C's **root**, which has **number of strong neighbor** as 4. B is C's **parent**, which has number of strong neighbor as 3. C itself has number of strong neighbor as 1.

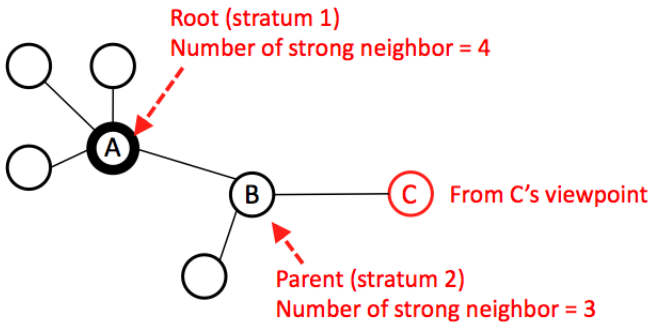


Figure 5.3: Definitions (from C's viewpoint): A is root with 4 strong neighbors. B is parent with 3 strong neighbors

Algorithm 1 describes the algorithm of parent selection. In initializing phase, each node sets itself as root, its stratum as 1, and root ID and parent ID are set as its own ID. During t_{hb} interval, node will get message from its neighbors. Node calculates sender's LET, and if the value is greater than LET_{bound}, node calls sender as strong neighbor, and allows this neighbor for next consideration.

The criteria for parent selection start from comparing candidate's root's strong neighbor with node's strong neighbor. This is done by the concept of selecting the highest density area of strong neighbor as the center of the system. The node with the highest density of strong neighbor tends to be the most reliable node, and the probability to get the rare change topology is higher. If sender's root's strong neighbor is greater than it root's strong neighbor, node will select sender node as parent node. However, in case that the comparison result is equal, the next criteria is done by comparing senders' strong neighbor,

stratum, and ID respectively.

As [JJ03] claimed that, the variance of the synchronization error increases along each branch of the tree as a linear function of number of hops, selecting the lowest stratum node can optimum tree by minimizing tree depth, thus, the overall synchronization error will get decrease. Last, in case that all previous criteria cannot be applied, comparing node ID can guarantee that nodes eventually decide on a parent node. Criteria for setting new parent are summarized below. Note that the criteria must be considered respectively in the mechanism:

CT1 New number of root's strong neighbor is greater than holding one

CT2 New number of root's strong neighbor equals the holding one and new number of parent's strong neighbor is greater than holding one

CT3 New number of root's strong neighbor equals the holding one and new number of parent's strong neighbor equals the holding one and new parent's stratum is lower than the holding one

CT4 New number of root's strong neighbor equals the holding one and new number of parent's strong neighbor equals the holding one and new root ID is less than the holding one

Algorithm 1 Parent selection

Require: received MPTP message
  if $LET > LET_{bound}$ **then**
    Update reference node's information when any of following conditions is true (consider orderly):
      - message.root_id equal my.root_id AND message.root_update_sequence is geater than my.root_update_sequence
      - CT1 (root_strong_neighbor)
      - CT2 (parent_neighbor)
      - CT3 (stratum)
      - CT4 (ID)
  end if

5.1.4 Constructing Hierarchical Topology

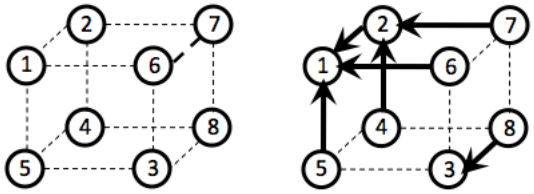
In this section, we explain some sample cases from implementing Algorithm 1. The purpose of protocol is to construct one hierarchical topology from the connected nodes. The basic case with two nodes, and the special case in joining two trees in cubic topolgy, where every node in the system has the same number of strong neighbor, are considered in sample cases.

CASE 1 (2 Nodes)

Considering a scenario with two nodes on Figure 5.4. Each node has unique ID, 1 and 2. When both come within LET_{bound} range, selecting parent process starts. CT1, CT2 and CT3 are invalid due to the same number of strong neighbor and stratum. However, CT4 is valid and, thus, topology is constructed with one decided root at node 1.

Figure 5.4: Parent selection between two nodes: Node 2 selects node 1 as root

CASE 2 (Joining tree)



(a) Starting system (b) Two separated trees

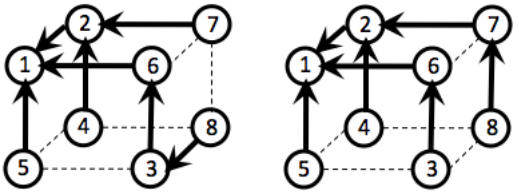
Figure 5.5: Cubic topology

Considering a cubic topology in Figure 5.5 (a) where every node has the same number of strong neighbor. By using CT1, CT2,and CT3, we get two separated trees as shown in Figure 5.5 (b). One has node 1 as root. Another has node 3 as root.

CT1 and CT2 cannot be applied because every node has the same number of strong neighbor. Moreover, CT3 cannot be applied because node 3 and 8 have lower stratum than node 1's children. However, by applying CT4 to node 3 and 8, they eventually join node 1 tree. Therefore, we get one tree in the system, where every node synchronizes its clock due to node 1's clock, as the root of topology. The possible results are given in Figure 5.6.

5.2 Broadcasting Time Synchronization Message

MPTP broadcasts time synchronization message in every t_{hb} time interval. Message is used both for topology construction by carrying senders information as described in Section 5.1, and for doing time synchronization similar to SNTP manner.



(a) (b)

Figure 5.6: Possible results after join the trees: (a) Hierarchical topology with 4-level height; (b) Hierarchical topology with 3-level height

5.2.1 Message

MPTP message consists of three parts: topology construction information, time request information, and time synchronization information. Figure 5.7 shows MPTP message format, and Table 5.1 presents attributes' description.

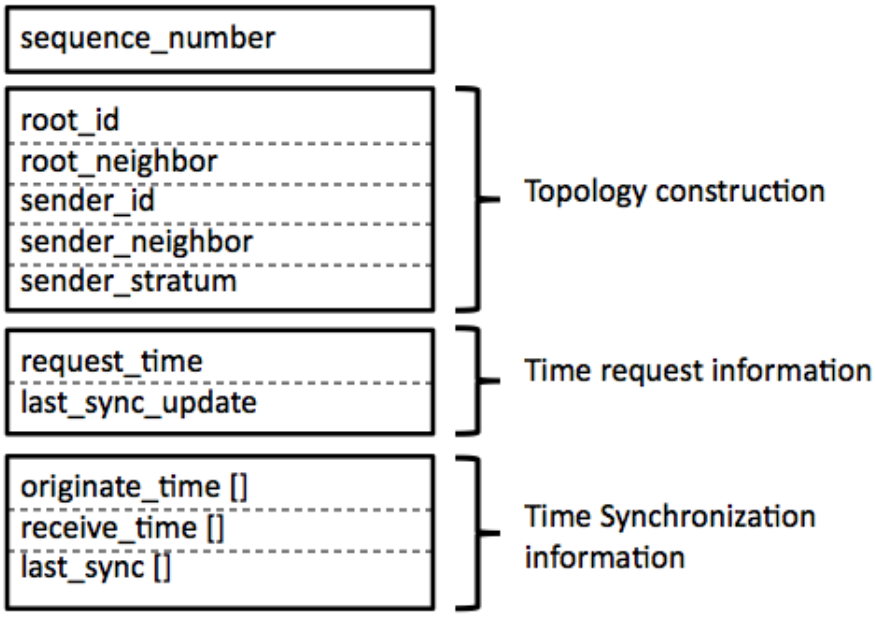


Figure 5.7: MPTP message format

Table 5.1: MPTP message format and attributes' description

Attribute	Number of bit	Description
sequence_number	16	Message sequence number
root_id	8	Sender's root ID
root_neighbor	8	Number of sender's root strong neighbor
sender_id	8	Sender ID
sender_neighbor	8	Number of sender's strong neighbor
sender_stratum	8	Sender's stratum
request_time	64	Message originate time for next round synchronization
last_sync_update	16	Logical clock depicts the last synchronization time. This field is used to prevent using old data for adjusting clock. Value increases in every clock adjustment.
originate_time[]	64 x []	Received *request_time* from nodes during last interval (number of bit varies due to number of received message during last interval)
receive_time[]	64 x []	Time at receiving message from nodes during last interval (number of bit varies due to number of received message during last interval)
last_sync[]	16 x []	Received *last_sync_update* from nodes during last interval. If value matched with receivers *last_sync_update*, allow time adjustment (number of bit varies due to number of received message during last interval)

5.2.2 Message Exchange Mechanism

Figure 5.8 shows one time synchronization process cycle. Assume that p_1 and p_2 have the same parent node, p_0. At t_{11} and t_{21}, p_1 and p_2 set request time to t_{11} and t_{21}, and broadcast message to the system. p_0 receives message from p_1 at t_{01}, and p_2 at t_{02}. p_0 keeps data until p_0 broadcasts message to system when start new t_{hb} interval.

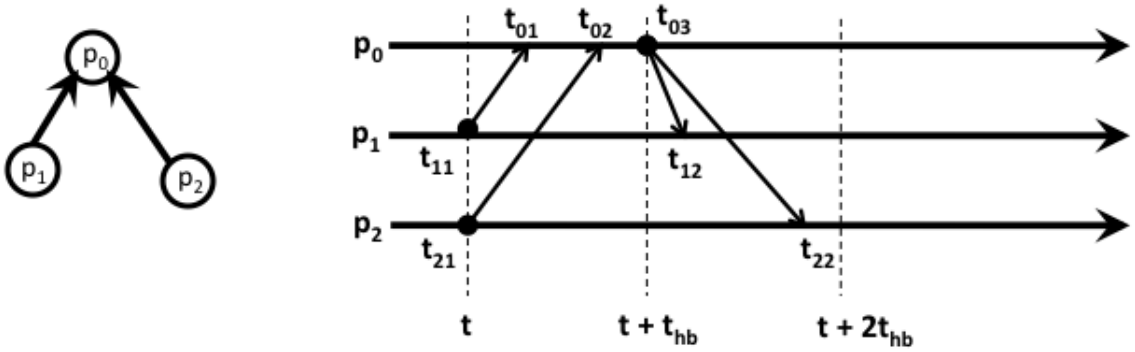


Figure 5.8: Message exchange mechanism

Thus, p_1 uses t_{11}, t_{01}, t_{03} and t_{12} to adjust the clock. Also, p_2 uses t_{21}, t_{02}, t_{03} and t_{22} to adjust the clock in the same manner as SNTP (presented in Section 2.2.2). p_1 and p_2 calculate delay and offset as follow:

$$d_{p_1} = (t_{12} - t_{11}) - (t_{03} - t_{01}) \tag{5.2}$$

$$t_{p_1} = \frac{(t_{01} - t_{11}) + (t_{03} - t_{12})}{2} \tag{5.3}$$

$$d_{p_2} = (t_{22} - t_{21}) - (t_{03} - t_{02}) \tag{5.4}$$

$$t_{p_2} = \frac{(t_{02} - t_{21}) + (t_{03} - t_{22})}{2} \tag{5.5}$$

Note that, every node will keep repeating this mechanism. One message acts both clock requesting and clock replying. Every request message will be replied, but receiver itself will decide which node information should be used.

5.3 Cycle

Cycle is possible to occur in the system by using MPTP. For instance, in the situation which the number of strong neighbor in two connected nodes changes frequently, and then cause the parent-children role switch alternately. This kind of situation leads to the cycle of data as shown in Figure 5.9, and may result in fluctuation of clock in the system.

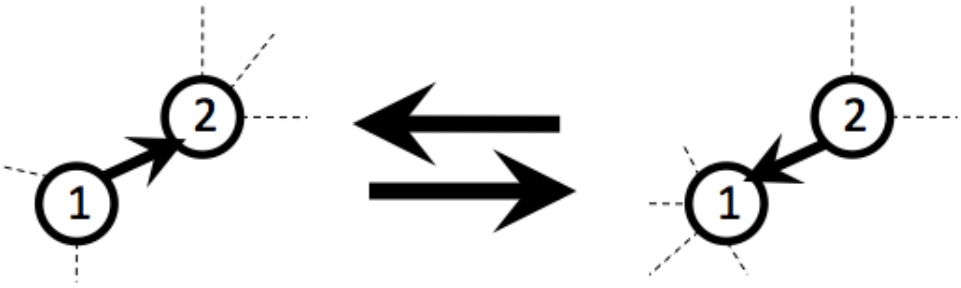


Figure 5.9: Cycle: Parent-children role switches alternately between node 1 and 2

Chapter 6

Population-based Time Protocol

In this chapter, we introduce a new scheme for time synchronization protocol. The protocol's idea is adapted from population protocol; an opportunistic protocol, which concentrates on an interaction between any two nodes.

6.1 Population Protocol

Population protocol model describes a collection of mobile agents that interact with one another to carry out a computation. According to [JE07], population protocol model was first inspired by work on trust propagation in a social network. The motivation given for the current model was the study of sensor networks in which passive agents were carried along by other entities; e.g. the sensor attached to a flock of bird. The name of model was chosen by analogy to population process in probability theory. Typically, the defining features of the basic model are:

- The system consists of a large population of indistinguishable finite-state agents.
- Instead of sending message or sharing memory, interactions between pairs of agents cause the two agents to update their state. The information flows in one direction only. A receiver learns the state of sender, but the sender learns nothing about the state of receiver.
- The interaction pattern is unpredictable
- Population protocol generally cannot detect when they have finished; instead, the output are converged after some finite time to a common value.

The Basic Model

The model consists of n agents, where $n \geq 2$. Each agent is given a finite input Σ. A **configuration** of system is described by a vector of all the agent's states. Start from initial configuration, an **interaction** between pairs of agents will change its state as a

result of the interaction. For example, when agents in state q_1 and q_2 meet and have an interaction, they will change into state q_1' and q_2' if $(q_1, q_2) \rightarrow (q_1', q_2')$ is transition relation. An **execution** of protocol is an infinite sequence of configurations $C_0, C_1, C_2, ...$, where C_0 is an initial configuration and $C_i \rightarrow C_{i+1}$ for all $i \geq 0$. Besides, the order in which pairs of agents interact is unpredictable. At any point of execution, each agent's state determines its output at the time. An agent in state q will give output value as $\omega(q)$.

6.2 Algorithm Description

In the model, we call agent as *node*. Nodes have unique identification and they are uniform, which mean all of them execute the same algorithm.

Node's clock can be seen as node's state. Starting from arbitrary state, each node initiates its own clock differently. Once any two nodes get a chance to meet each other, they interact in order to update their states. Receiver, by defined as node with less ID than another, will adjust its clock by taking **average** between sender's and its own clock. Note that sender (node with greater ID) will not change the state after interaction. Algorithm 2 describes process mechanism.

Assumption 1. *All clocks in the system run at the same rate, but are initiated with different offset.*

Assumption 2. *Every node interacts infinitely in the execution.*

Assumption 3. *Every node in the system is correct node.*

Definition 1 (Sender-Receiver)**.** *During the interaction, node with greater ID will act as sender, while one with less ID will act as receiver. Only receiver will adjust clock after interaction.*

Definition 2 (Leader Election)**.** *In the system, a node with the greatest ID is a leader, which provides system a clock reference. Leader is sender in every interaction.*

Definition 3 (Legitimate Configuration)**.** *The system executing algorithm is in legitimate configuration if all clocks have the same value as leader's clock.*

Algorithm 2 POP-B's time synchronization

```
if message.node_id > my.id then
    my.clock = (message.clock + my.clock)/2
end if
```

Lemma 1. *Let e be an execution of Algorithm 2 starting in an arbitrary configuration. e eventually reaches legitimate configuration.*

Proof. Starting the system under Assumption 1, node adjusts its clock offset to the node with greater ID in every interaction as described in Definition 1. Since the interaction is iterated infinitely and all clocks in the system run at the same rate, all nodes clock offset will eventually converge to leader's clock. Thus, the system will reach legitimate configuration. □

Lemma 2. *By applying Algorithm 2, system will remain in legitimate configuration forever once it enters legitimate configuration.*

Proof. Consider the system under Assumption 3 ,and since every node interacts infinitely, all clocks in system will keep synchronized with the correct clock value. Therefore, the system will remain in legitimate configuration forever. □

Chapter 7

Performance Analysis

MPTP and POP-B were implemented on OMNET++ [OMNET] to evaluate protocol performance. The evaluation was done based on comparing against SNTP and RTSP. We describe the details for simulation model, evaluation metric, and result in this chapter.

7.1 Simulation Model

To evaluate the proposed MPTP and POP-B in mobile sensor network, we conduct OMNET++ by considering different node's speed. The simulation area is 100m x 100m square region whereby nodes are placed randomly inside the area. The mobility model is random waypoint torus. Once the simulation begins, each node moves toward a randomly selected location with a fixed speed that is based on speed type as aforementioned. When node moves hit the bound, it will be placed at the opposite bound and continues moving to destination. The number of node is 20 nodes. Nodes transmission range is 10 m. The simulation time is 30,000 seconds.

7.2 Evaluation Criteria

The performance metrics that are used in this simulation are defined as follow:

1. Network synchronization error
   Network synchronization error is defined as the average of time different in every node pairs in the system.

2. Convergence time
   The convergence time is defined as the time needed by the system to converge to a desired Network synchronization error.

Table 7.1: Average network synchronization error compared among four protocols in node without mobility

Protocol	Network Synchronization Error_{avg} in clock with no drfit but offset (ms)	Network Synchronization Error_{avg} in clock with drfit and offset (ms)
SNTP	2.27000	4.13230
RTSP	4611.08	22178.79260
MPTP	1.39791	3.10633
POP-B	1.40845	2.53125

7.3 Simulation Result

7.3.1 Network Synchronization Error

Static Sensor Network

We first evaluate SNTP, RTSP, MPTP and POP-B's network synchronization error in node with no mobility system. In the simulation, we place 25 sensor nodes in 5x5 grid topology. The space between any two nodes is 8 m. The simulation time is 30,000 seconds. The simulation is done by observing two node's clock conditions; clock with no drfit but offset, and clock clock with drfit and offset.

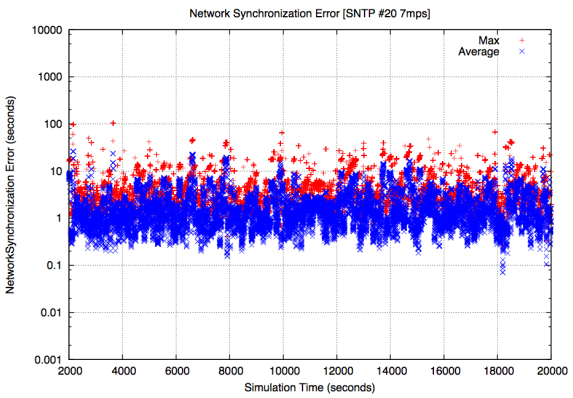
According to Table 7.1, we observe that MPTP and POP-B obtain almost the same performance in static conditon. However, SNTP performs in bigger synchronization error result compared to MPTP and POP-B due to the message collision problem since SNTP requires to reply message suddenly in each operation, which causes the number of message in the system becomes double compared to MPTP and POP-B. Similarly, since RTSP needs 4 messages to establish the topology and operate time synchronization process, RTSP performs worst in synchronization error result among 4 protocols for both clock with no drift but offset condition, and clock with drift and offset condition.

Mobile Sensor Network

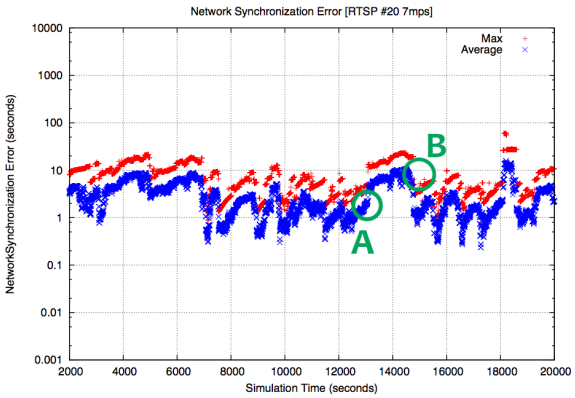
Figure 7.1 (a), (b), (c) and (d) show SNTP, RTSP, MPTP and POP-B's network synchronization error at node speed 7 m/s respectively. Since the result is the same through out the simulation run, so we capture some duration in order to present more clearly in detail.

In Figure 7.1 (a), SNTP results in average error at 2.1914045 seconds and maximum error is 4.60847921 seconds. At some time, the system may get some peak of error because some nodes separated from group and did not get synchronized for some amount of time.

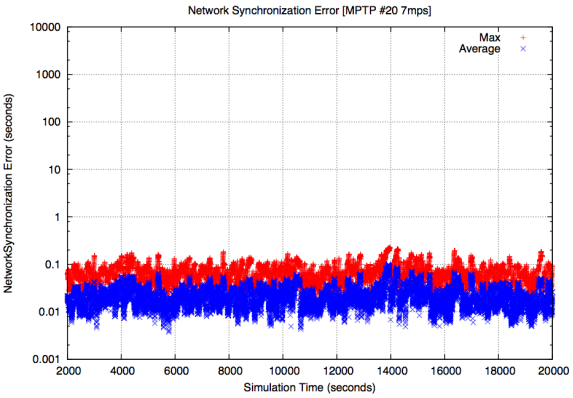
Figure 7.1 (b) presents RTSP's network synchronization error. As we can see from the graph, RTSP cannot tolerate mobility since the error oscillates and cannot maintain in the certain value. An average network synchronization error is 4.52858891 seconds and maximum error is 6.6989388 seconds. The interesting cases are marked at point A and B on the graph. The condition that error starts rising up gradually at point A happens since



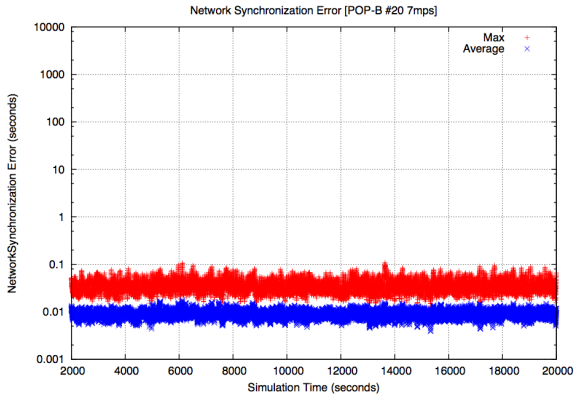
(a) SNTP ($Error_{avg} = 2.1914045secs$) (+/-0.013 s w/99% conf)



(b) RTSP ($Error_{avg} = 4.52858891secs$) (+/-0.093 s w/99% conf)



(c) MPTP ($Error_{avg} = 0.023677secs$) (+/-0.057 ms w/99% conf)



(d) POP-B ($Error_{avg} = 0.009016secs$) (+/-0.007 ms w/99% conf)

Figure 7.1: Network synchronization error (Number of node = 20; Random waypoint torus - Node speed = 7 m/s)

a node in the system lost its connection with its reference, and stucks in being absent from clock synchronization from two reasons:

1. Node is separated from topology.

2. Node tries to establish connection with other nodes in its candidate list, however, the information is totally out-of-date due to the frequently change in topology.

The error increase rate depends on the number of node that become unsynchronized. The number of data miss in candidate list and time separated from topology affect the duration of error rising. The synchronization for unsynchronized nodes occurs at point B in graph, thus, causes the error drops in the system. The error decrease rate depends on the speed of spreading one reference clock to unsynchronized nodes.

According to the result, RTSP performs worse performance than STNP because RTSP's candidate list itself became the weak point in sparse and highly mobile system. RTSP gets

Table 7.2: Network synchronization error compared among four protocols

Protocol	Network Synchronization Error$_{avg}$ (seconds)	Network Synchronization Error$_{max}$ (seconds)
SNTP	2.1914045	4.60847921
RTSP	4.52858891	6.6989388
MPTP	0.023677	0.071458
POP-B	0.009016	0.036307

high average error and cannot reach stable state since the protocol always uses the out of date candidate in the list, moreover, the process to construct and do synchronization takes too much time, thus, the node cannot synchronize with parent before connection lost.

Figure 7.1 (c) shows MPTP's network synchronization error. MPTP achieves 0.023677 seconds in average error, and maximum error at 0.071458 seconds. According to all experiments, the cycle does not affect the system clock stability as there is no clock fluctuation in MPTP's result.

Figure 7.1 (d) depicts POP-B's network synchronization error. POP-B achieves the very low error, with 0.009016 seconds. Furthermore, by using POP-B, system gains low maximum error at 0.036307 seconds.

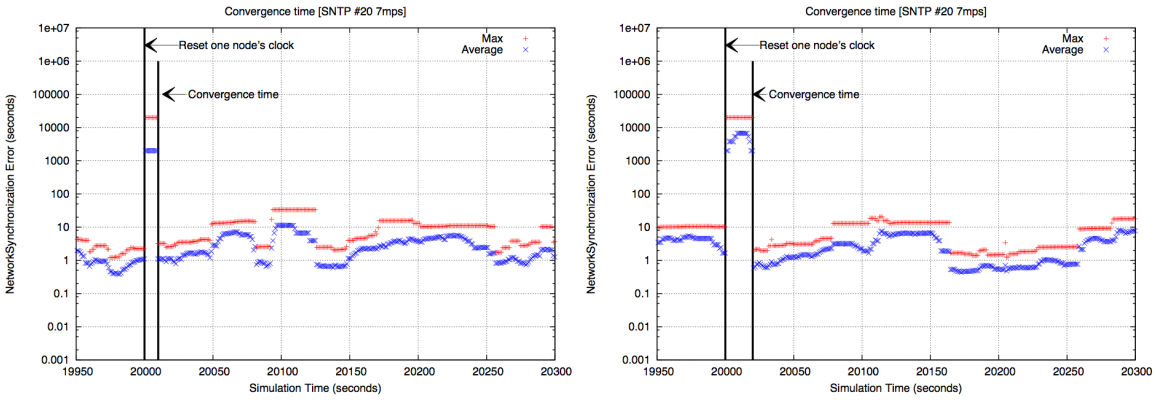
Consequently, comparing among four protocols, MPTP performs at 93 and 191 times better clock accuracy than SNTP and RTSP, while POP-B performs at 243 and 502 times better clock accuracy than SNTP and RTSP respectively. Without constructing the topology, POP-B lets nodes in the system synchronize with each other freely while focusing on converging their clocks into one same direction. Thus, by using the advantage of node's high mobility, the opportunity to make synchronization increases and leads to more clock accuracy in the system. For MPTP, since protocol tries to construct the strong connection topology and adds the ability for dynamically change parent, MPTP can almost achieve the stable system and same average error as in POP-B. However, by strict on hierarchical synchronization, the protocol leads to some jumping error peaks when a node lost its connection with previous parent and spends time on constructing new connection. SNTP gives the performance almost the same manner as MPTP, however, without dynamically parent update mechanism, SNTP faces the difficult problems with connection loss and lost synchronization gap time, thus, SNTP gains even higher error average than MPTP. Table 7.2 concludes the network synchronization error among four simulated protocols.

7.3.2 Convergence Time

To measure the deviation from global average time, we evaluate convergence time performance among four protocols by conducting one node in the simulation to reset its clock at simulation time 20,000 second to 0. Convergence time was recorded from the time node reset itself and system gained a high jump of error until the system reaches and remains

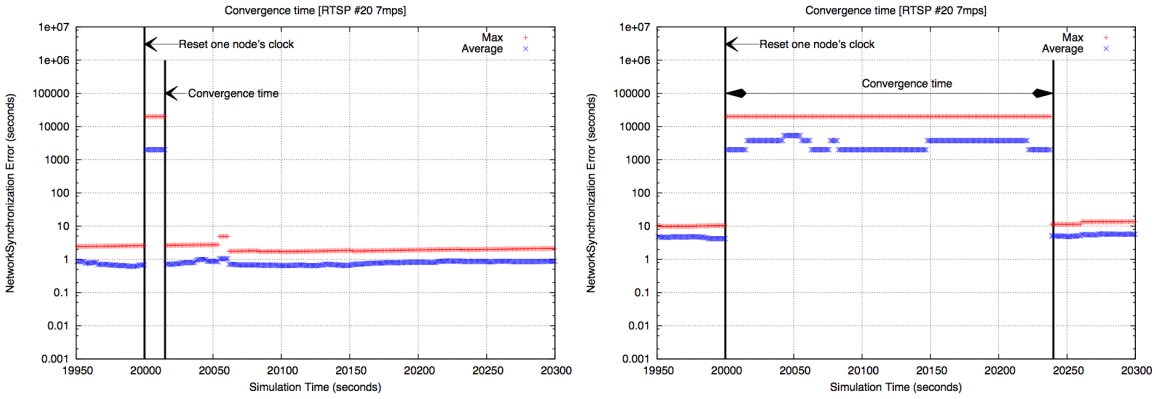
in stable state again.

Figure 7.2, Figure 7.3 and Figure 7.4 depict the convergence time in SNTP, RTSP and MPTP respectively. For these three hierarchical protocols, the amount of time used for recovering the system is various and depends on node's condition at the time. The best case is shown in (a), where the node can recover its clock in its next synchronize interval. In this case, the reset node still have connection within the topology and the topology's formation is still not broken, therefore, after the reset time, node can immediately resynchronize with its old parent. The common case is presented in (b). After node was reset, error jumps high and takes amount of time to become stable again. the situation can be described from following two reasons:



(a) Best case (b) Common case

Figure 7.2: SNTP's Convergence time

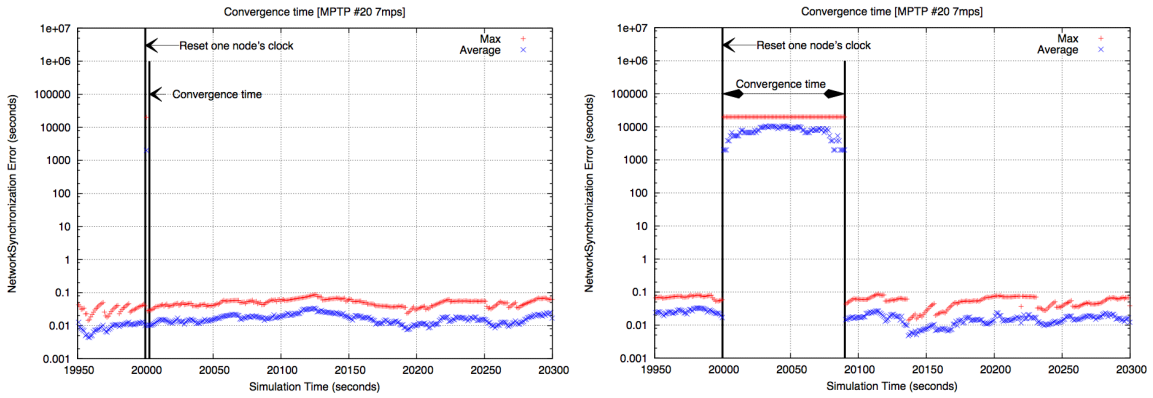


(a) Best case (b) Common case

Figure 7.3: RTSP's Convergence time

1. The reset node is separated from topology.

2. The reset node, as a parent for higher stratum nodes, gives its new reset clock to the system. This time spreads throughout the remaining higher stratum nodes and



(a) Best case (b) Common case

Figure 7.4: MPTP's Convergence time

causes system clock fluctuation in amount of time. Thus, protocol requires a lot of time to recover the entire system. We can see the the effect of this situation in Figure 7.2 (b), Figure 7.3 (b) and Figure 7.4 (b) as the error still increases for a while after node reset the clock, and gradually decreases from adjusting the affected nodes before a sudden drop to stable state.

Figure 7.5 depicts the convergence time in POP-B. Unlike SNTP, RTSP, and MPTP, POP-B gives the result in one pattern. Since the protocol calculates the average clock for adjustment, once the error rises after node reset, the error gradually decreases until system become stable again.

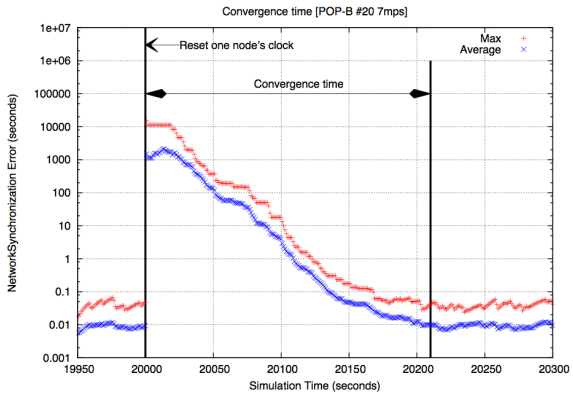


Figure 7.5: POP-B's Convergence time

The average and maximum convergence time conclusion are presented in Table 7.3. SNTP and MPTP achieve the lower average convergence time compared to POP-B and RTSP. Since hierarchical protocol maintains level structure in topology, adjusting the reset clock affected nodes can be either immediately done, or can take amount time to adjust all affected nodes. On the other hand, POP-B does not maintain any structure

Table 7.3: Convergence time compared among four protocols

Protocol	Convergence Time$_{avg}$ (seconds)	Convergence Time$_{max}$ (seconds)
SNTP	8.5	26
RTSP	309.1	904
MPTP	18.1	89
POP-B	120.05	169

like SNTP and MPTP, instead, protocol speads the clock throughout the entire system, and thus, POP-B always takes time to stabilize the entire clocks. Last, RTSP is also hierarchical protocol, however, convergence time is slow since protocol has problem with out-of-date reference node's information.

7.3.3 Effect of Changing Node Speed on Performance

Figure 7.6 shows the performance of four types of protocol varying with the node's speed with value at 99% confidence interval. RTSP's synchronization error grows when node moves faster and entire error starts to diverge earliest among four protocols. SNTP's synchronization error grows at the same manner as RTSP but in the slower rate. In the first period, RTSP seems to perform in better accuracy than SNTP. But the breakpoint is at 8 m/s, where SNTP tends to perform better than RTSP. SNTP and MPTP, as hierarchical protocol, have the limitation the same speed at around 2,000 m/s. For MPTP and POP-B, both protocols give the better performance when node speed increases, and POP-B performs the synchronization in the lower error and variance compared to MPTP. However, MPTP's synchronization error starts growing up at speed 8 m/s while POP-B's synchronization error starts at 512 m/s. POP-B starts to diverge latest among four protocols. As a result, we can see that both MPTP and POP-B can tolerate mobility much better than SNTP and RTSP, moreover, POP-B gives the best performance in this experiment.

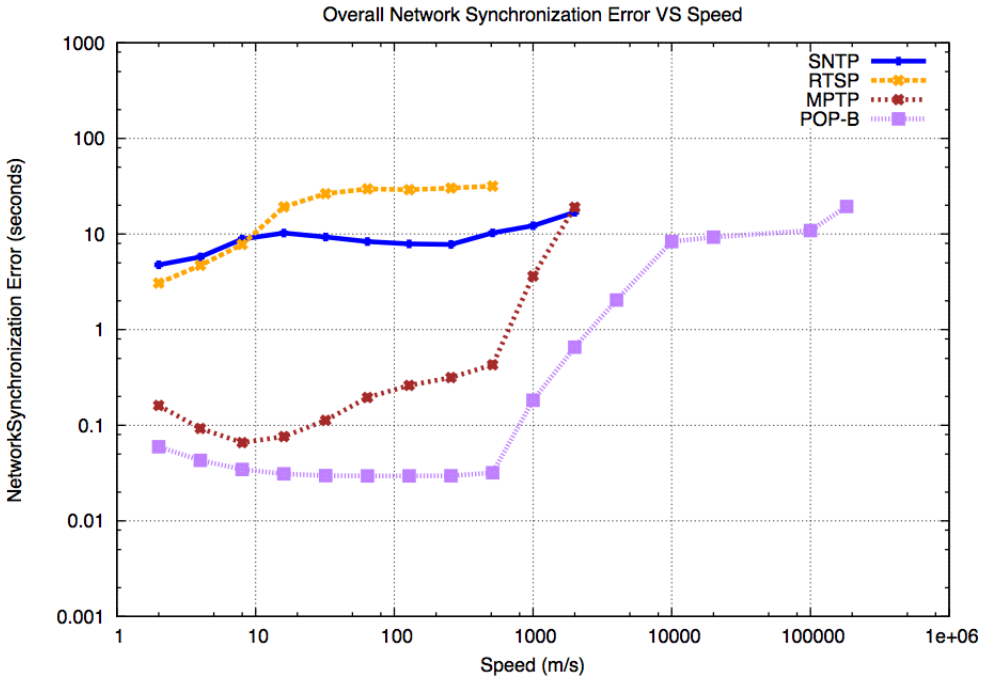


Figure 7.6: Network synchronization error versus node speed

Chapter 8

Conclusion

8.1 Conclusion

In our research, we address the problem of time synchronization in sparse and highly mobile sensor netwoks. As existing protocols still cannot handle with the frequent topology change caused by mobility, we propose two new protocols for tackling the problem.

First, Mobility Prediction Time Protocol (MPTP), which adapts the method of mobility prediction and proposed parent selection criteria so that protocol can construct the strong connection topology for rare topology change. Protocol allows topology reconstruction before the connection between two nodes breaks. Moreover, by using the proposed broadcasting mechanism, protocol can eliminate the overhead processes. Through experimental study based on simulations, our proposed protocol can perform time synchronization in high accuracy compared to SNTP and RTSP. MPTP is able to converge when some nodes in system reset its clock and performs even better when mobility speed increases. However, position system equipment is required to perform mobility prediction, trade-off between clock accuracy and power consumption should be concerned.

Second, we proposed Population-based Time Protocol (POP-B), which concentrates on spreading clock information over the system by adapting the concept from opportunistic scheme protocol. The experiment results indicate that POP-B can achieve the most accuracy time among four protocols. POP-B can perform even better when mobility speed increases. With the simple idea, non-structural and opportunistic scheme is more efficient than constructing solid connection idea, and can achieve a high degree of robustness in sparse and high mobility system.

To summarize, an up-to-date information maintenance mechanism is significant in order to increase the availability of clock reference node. Moreover, since available time to exchange clock information is short and rare due to the high mobility, proficiently communication time optimizing and clock information coverage processes should be mainly concerned in sparse and highly mobile network.

8.2 Open Questions and Future Works

For the open questions, we questions about investigating MPTP and POP-B's performance in different mobility models. In addition, enhancing POP-B to handle the system with Byzantine failure is also an interesting problem. A node that has a Byzantine failure may behave arbitrarily: it interacts with all other nodes and gives the wrong clock for each interaction. This kind of behavior causes the system to become unstable, thus, the next research may consider the method to validate the received clock information or the number of Byzantine nodes that protocol can tolerate.

Bibliography

[DCG10] D. Gavalas, C. Konstantopoulos and G. Pantziou, "Mobility Prediction in Mobile Ad Hoc Networks", in "Next Generation Mobile Networks and Ubiquitous Computing", S. Pierre (Ed.), ISBN10: 160566250X, Chapter 21, pp. 226-240, IGI Global, USA, June 2010.

[EE02] Elson, J., Estrin, D. (2002). "Fine-Grained Network Time Synchronization using Reference Broadcast.", The Fifth Symposium on Operating Systems Design and Implementation (OSDI), p. 147-163

[HB05] Hwang, S., Baek, Y., Reliable Time Synchronization Protocol in Sensor Networks Considering Topology Changes, Proceedings of the IFIP International Conference on Embedded And Ubiquitous Computing, 2005.

[HR00] H.G. Berns and R.J. Wilkes. GPS Time Synchronization System for K2K. IEEE Transactions on Nuclear Science, 2000

[IFWYE02] I.F. Akyildiz and W. Su and Y. Sankarasubramaniam and E. Cayirci, Wireless Sensor Networks: A Survey IEEE Computer, vol. 38, no. 4, pages 393-422, Mar. 2002.

[JE07] J. Aspnes and E. Ruppert. An introduction to population protocols. Bulletin of the European Association for Theoretical Computer Science, 93:98-117, October 2007. Columns: Distributed Computing, Editor: M. Mavronicolas.

[JD01] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. In 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Network and Mobile Computing, San Francisco, USA, April 2001.

[JJ03] J. van Greunen and J. Rabaey, Lightweight Time Synchronization for Sensor Networks, Proc. 2nd ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA), San Diego / CA, 2003.

[GKS03] Ganeriwal, S., Kumar, R., Srivastava, M. (2003). "Timing-Sync Protocol for Sensor Networks.", The First ACM Conference on Embedded Networked Sensor Systems (SenSys), p. 138-149

[MAR] Figure: http://mars.cs.umn.edu/projects/current/PerformanceCLATT/ PerformanceCLATT.html

[MKSL04] Maroti, M., Kusy, B., Simon, G., Ledeczi, A. "The Flooding Synchronization Protocol.", Proc. Of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2004.

[MROCHE06] M. Roche, Time Synchronization in Wireless Networks , April 2006. [Online]. Available: http://www.cse.wustl.edu/∼jain.

[NTP3] Network Time Protocol (Version3) Specification, Implementation and Analysis, http://www.faqs.org/rfcs/rfc1305.html.

[OMNET] OMNeT++, Discrete Event Simulation System, http://www.omnetpp.org

[SNTP4] Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, http://www.faqs.org/rfcs/rfc4330.html.

[WSM01] W. Su, S.-J. Lee, and M. Gerla, Mobility Prediction and Routing in Ad Hoc Wireless Networks, Intl J. Network Management, no. 11, vol. 1, Feb. 2001.