RELATIVE CLOCK DRIFT RATE BASED SECURE TIME SYCNRHONIZATION FOR WIRELESS SENSOR NETWORKS

by

JAE SUNG CHOI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2006

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude towards Dr. Yonghe Liu for giving me priceless guidance and invigoration. Without his patience and continuous help this thesis would not have been completed.

I am extremely thankful to the members of my thesis committee. I give deep appreciation to Dr. Sajal Das for serving the best environment for my research. I thank from the bottom of my heart to Dr. Ramesh Yerraballi for donating his precious time to ameliorate my thesis.

Words are insufficient express my best appreciation to my father and mother for their unqualified love and great support. Also I am very thankful to my brother for his encouragement and assistance.

I would like to give my best love and thank to my wife and my daughter. Without their unconditional and endless love, my work would not have been possible. I dedicate this work to my family.

April 5, 2006

ABSTRACT

RELATIVE CLOCK DRIFT RATE BASED SECURE TIME SYCNRHONIZATION FOR WIRELESS SENSOR NETWORKS

Publication No. _____

Jae Sung Choi, M.S.

The University of Texas at Arlington, 2006

Supervising Professor: Yonghe Liu

Time synchronization is critical issue to many wireless sensor network applications such that target tracking, TDMA radio scheduling, and secure localization. However, the most of existing time synchronization algorithms in wireless sensor networks did not consider malicious attacks in hostile environments. In this thesis, we propose a *Relative Clock Drift Rate Based Secure Time Synchronization* (RSTS) schemes to address security problems. RSTS alleviates delay attacks and incorrect time stamp transmissions caused by external or internal malicious attackers. We discuss a simple estimation technique to calculate a relative clock drift rate between sender-receiver clocks, because the estimated relative clock drift rate is important to detect malicious attacks, when we employ a threshold-based algorithm to calculate bounds of

acceptable the offset. We show that RSTS incurs minimal computational overhead. Further, it does not consume any extra bandwidth when it is compared to any other sender-receiver synchronization algorithms.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Secure Time Synchronization in Wireless Sensor Networks

Time synchronization is critical issue to many wireless sensor network applications such that target tracking [16], uTESLA for authenticated broadcasting [17], [18], TDMA radio scheduling [19], and localization [20], [21]. And all of these the wireless sensor network applications are designed under the assumption of accurate time synchronization. Therefore if the sensor network does not prepare the accurate time synchronization protocol, most of sensors can not work correctly with their own objects. In wireless sensor network, the applications of sensors are used own physical clock which oscillates frequencies. Under ideal world, the oscillator generates consistent frequency without any drift, and all of physical clocks report exactly same time value relative to the true time without any time difference. In real world, every physical clock accounts different times relative to the true time because of various sources of clock drift such that frequency skew, influence of temperature, and ageing of oscillator [11], [12]. Since, sensor's physical clocks have different clock drift, accurate clock synchronization protocol is necessary in wireless sensor networks for satisfaction of the assumption about time synchronization in the other sensor applications. Currently, there are outstanding time synchronization protocols which service synchronization with time error within microseconds, such as RBS [1], TPSN [4], and Tiny-Sync [8].

However, most of existing time synchronization protocols are not designed against malicious environments. Privation of secure mind in time synchronization raises the wireless sensor's abnormal operations caused by hostile time attacks. For example, the application for localization determines the deployed sensors' location with the uses of time-of-flight of sound, the reported location is not trustable under malicious time attack. In time synchronization for sensor networks, an external or internal malicious adversary can attack several ways such as passive eavesdropping of timestamps and active masquerading, replay of timestamps, and modification of timestamps.

In operation of secure time synchronization, a timestamp has to keep following security characteristic; Authentication, integrity, and freshness. Authentication and freshness of time stamps can be assured by current secure mechanism such as the use of shared secret key, Message Authentication Code (MAC), and a random nonce. Also these mechanisms provide message integrity against external attackers. But the mechanisms can not protect against modification of timestamps caused by internal compromise node. Since the wireless sensors can be deployed in hostile environments like a battle field, multiple deployed sensors can be captured and programmed malicious codes and then the sensors are compromised and replaced at the original position. Internal malicious nodes will affect time adjustment using intentional interferences such as a delay attack and transmission of incorrect time message. In sensor networks, these kinds of internal time attacks bring about distribution of incorrect time among the networks.

Since each sensor' clock is working with inherent drift rate, the offset between two relative sensor clocks is getting longer with time goes by even the sensors are initiated at same moment. Without adjustment of time, the sensor applications can not operate correctly in wireless sensor networks because of accumulated offset. Ideally, if the sensor can estimate its own actual clock drift rate by itself, the sensor exchanges information of the actual drift rate each other for estimating the relative clock drift rate between two sensors. Using actual relative clock drift rate, the sensors can obtain various benefits such as self-time-adjustment, minimal consumption of energy resource and self-defense against multiple time attacks. But there is no way to estimate the actual clock drift rate by itself. Instead, in this thesis, we try to estimate the relative clock drift rate between two nodes under given conditions. Then we focus on how to use the estimated drift rate against malicious time attack.

1.2 Contributions

First, we propose a simple protocol for secure time synchronization against possibility of malicious compromise node's modification of timestamps. Existing time synchronization protocols do not defend against harmful mechanisms of internal adversary. Relative Clock Drift Rate based Secure Time Synchronization (RSTS) protocol which is based on sender-receiver synchronization method, is efficient detecting mechanism of timestamps modification caused by internal adversaries. The results of our experimental implementation with the uses of MICA2 motes shows 76.6% of detecting performance against time attack with 100 microseconds, and 100% of detection performance when compromise node time attacks with over 1 millisecond.

Second, our RSTS incurs minimal computational overhead. Further, it does not consume any extra bandwidth when it is compared to any other sender-receiver synchronization algorithms, because RSTS does not require any extra communication with a relative node or a base station.

1.3 Thesis Overview

In the remainder of this thesis, we will study basic concepts for time synchronization and existing time synchronization protocols in detail. Also we claim the secure problems in the existing time synchronization protocols, and then we address our new secure time synchronization protocol against compromise nodes' attack. Finally, we describe the result of experimental implementation.

Chapter 2 reviews backgrounds of time synchronization protocols such as clock terminology and universal concepts of time synchronization. Also we explain the various sources of time errors in the wireless sensor networks.

Chapter 3 addresses related work in time synchronization for wireless sensor networks. And we claims secure problems of the related works in detail.

In Chapter 4, we present our Relative Clock Drift Rate based Secure Time Synchronization (RSTS) protocol against malicious attacks caused by compromise nodes. In this chapter, we show how to estimate boundary of acceptable offsets with the uses of given conditions.

In Chapter 5, we describe the result of experimental implementation with the uses of MICA2 motes. The results show the detecting performances of our RSTS under various conditions.

In Chapter 6, we summarize our secure time synchronization protocol and address our conclusions and future works in research of secure time synchronization areas.

CHAPTER 2

BACKGROUNDS OF TIME SYNCHRONIZATION PROTOCOLS

In this chapter, we address background of time synchronization protocols in sensor networks. We explain the basic clock terminology and notations for this thesis in Section 2.1. Then the universal concepts of time synchronization are described in Section 2.2. The computer time synchronization algorithms' characters are explained in this section. Finally, the sources of time errors in the wireless sensor network are described in detail in Section 2.3.

2.1 Clock Terminology

In sensor networks, a physical clock is equipped on each sensor device. A physical clock uses an electronic oscillator which generates systematic frequencies by quartz crystal. An ideal true clock reports true time t at anytime. When the true clock reports at time t, a sensor A's physical clock announces local time caused by function $C_A(t)$. Clock offset which is the time difference between two clocks at specific true time t, is $C_A(t) - C_B(t)$. Clock skew which is the difference in the frequencies of two clocks, is $C_A^{'}(t) - C_B^{'}(t)$. The maximum clock drift rate ρ of faultless clock is addressed 10^{-6} in a current hardware clock which is given by the hardware manufacture. If the clock skew is bounded by the maximum drift rate ρ, clock values are allowed to diverge at a rate in the range of 1- ρ to 1+ ρ [10]. Clock drift is occurred several causes such that hardware clock's frequency skew, oscillator's ageing, and variation of

temperature [11]. Clock drift rate is the difference in the precision between a clock and the true clock [2]. If a physical clock A's clock drift rate is equal to one, this clock is perfectly working as true clock. Since, typical sensor nodes are installed cheap hardware clocks, the most of clocks' reported times are faster or slower than true time which means clocks' drift rate is greater or less than one. Similarly, we consider a relative clock drift rate between two physical clocks which are not the true clocks, the relative clock drift rate can be equal to one, or greater or less than one. If the relative clock drift rate is equal to one, two clocks have same clock drift rates. The other hand, if the relative clock drift is no equal to one, one clock works faster or slower than the other clock. Figure 2.1 illustrates the behavior of different two clocks.

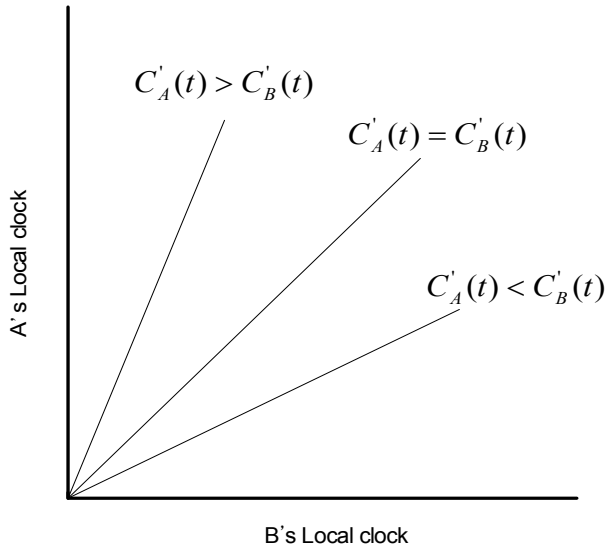

Figure 2.1 Behavior of different two clocks

Table 2.1 Notations

t	The reported true time by the true clock
$C_i(t)$	Local clock time at node *i* when the true time is *t*
$C_i^{'}(t)$	The frequency of C_i at the true time *t*
ρ	Maximum drift rate of non-faulty clock
θ	The clock offset relative to the reference clock
Δ	Time error caused by uncertainty of packet delay
ppm	Parts Per Million(10^{-6})
$\alpha_R^{i\to j}$	The real relative clock drift rate between node *i* and *j*
$\alpha_E^{i\to j}$	The estimated relative clock drift rate between node *i* and *j*
$RD_{t1\to t4}^{i\to j}$	The relative clock drift between the node *i* and *j* from real time t1 to t4
interval	Re-synchronization interval
θ_R	The real offset of a synchronization requester node relative to the reference node
θ_C	The calculated offset of a synchronization requester node relative to the reference node using clock offset-delay estimation method
$\Delta_{\max}$	The expected maximum time error cause by uncertainty of packet delay
$\Delta_{\min}$	The expected minimum time error cause by uncertainty of packet delay
S_i	The summation of send time, access time, and transmission time at node *i*
R_i	The summation of reception time, and receive time at node *i*
$P_{i\to j}$	Propagation time from *i* to *j*
DR^i	Node *i*'s clock drift rate relative to real time

2.2. Universal Concepts of Time Synchronization

Since time synchronization is critical issue in distributed systems, there are a lot of time synchronization protocols for computer networks. Mainly we can classify time synchronization schemes to the clock offset-delay estimation method, the time transmission method and the set-valued identification method by characters of the time adjustment operations. Also the synchronization schemes are classified to Sender-Receiver synchronization and Receiver-Receiver synchronization approaches.

2.2.1 Clock Offset-Delay Measuring Method

The clock offset-delay measuring method is generally used on the Internet. This method is employed by the Network Time Protocol (NTP) [3] which is an Internet standard protocol for synchronizing a clock to some time reference servers. Also Cristian [6]'s Remote Clock Reading method is similar as the clock offset-delay measuring method. The clock offset-delay measuring method requires four timestamps for each clock adjustment operation. When node A which is requester of time synchronization, sends time synchronization request packet to node B which is reference node, at time T1. And B receives A's request packet at T2, then it sends ACK which includes timestamps T2 and T3, to A. After A received ACK from B at T4, A can calculates an offset relative to the reference node B with four timestamps used by simple equations like as,

$$Offset = \frac{(T2 - T1) - (T4 - T3)}{2},\ Delay = \frac{(T2 - T1) + (T4 - T3)}{2}$$

However, variation of the network traffic and routing delay reduce accuracy of the offset-delay measuring method, because the network traffic and routing delay is uncertainty factors and hard to calculate.

2.2.2 Time Transmission Method

The time transmission method is used in the Time transmission protocol (TTP) [7] which is used by a series of synchronization messages and message delay statistics. Node B, which is the time reference, sends a series of time synchronization message to node A, which is the target node. Each time synchronization message is contained the sending time according to the B's local clock. The target node records the arrival time of each time synchronization massage according to the own local clock. Figure 2.2 shows the sending and receiving the series of timestamps.

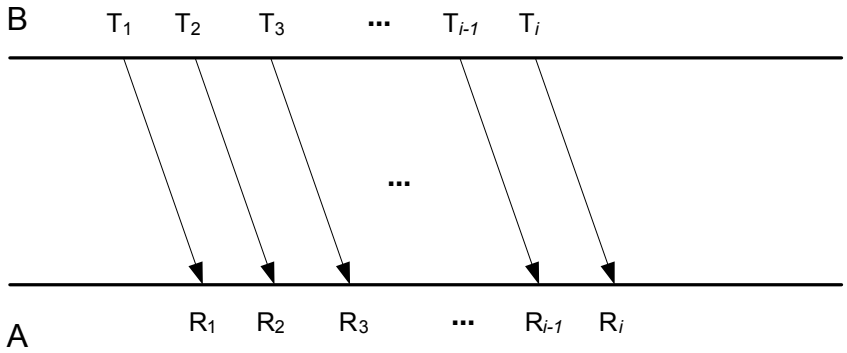

Figure 2.2 Time source node sends series of timestamps to target node

Then node A estimates the time source node's time using the following equation,

$$T_{est} = R_n - (\frac{1}{n}\sum_{i=1}^{n} R_i - \frac{1}{n}\sum_{i=1}^{n} T_i) + \bar{d},$$

where R_n denotes the receiving time of the n[th] message according to the target node, T_n is the sending time of the n[th] message according to the reference node, and $\bar{d}$ is the expected value of message delay. However, this method requires a lot of time synchronization messages for highly accurate time synchronization, and it occurs overheads of consumption of bandwidth and computation. Also the use of the expected value of message delay reduces the precision of establishment of time because of variation of message actual delays.

2.2.3 Set-Valued Identification Method

The Set-valued identification method [15], [22] is beneficial in network systems when the modeling uncertainty is not straightly captured by a previous models. When we assume that the distributed system consists of processors P_i for i = 1, ..., N, t_i denote the local time on the clock for processor P_i , we can derive linear equation for two relative processors P_i and P_j.

$$t_i = a_{ij} t_j + b_{ij}$$

Where a_{ij} expresses the relative skew and b_{ij} denotes relative offset between two physical clocks.

P_i sends N messages to P_j at P_i's local times $\underline{t_{ik}}$ for k=1,...,N. After P_j received the first message from P_i at P_j's local time tj1, P_j sends out a message to P_i included timestamp t_{j1}. The processor P_i receives timestamp at $\overline{t_{i1}}$. After P_j replies the series of N messages at t_{jk}, and P_i received series of reply messages at $\overline{t_{ik}}$, P_i has N sets of timestamps,

$$(\underline{t_{ik}}, t_{jk}\overline{t_{ik}})$$

With the uses of the set of timestamps, a graph is expressed the relative a_{ij} and b_{ij} using the slop and Y-intercept of line, as shown in Figure 2.3. However, the accuracy of set-valued identification method is affected seriously from message corruptions and losses.

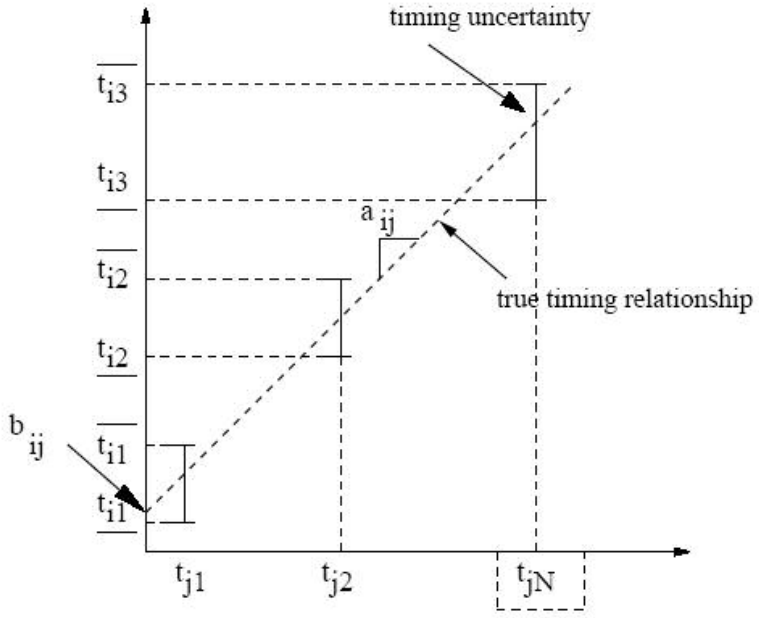

Figure 2.3 Set of triple timestamps expressed with the P_j's local time on the X-axis and the P_i's local time on the Y-axis [22]

2.2.4 Sender-Receiver and Receiver-Receiver Time Synchronization Approaches

Sender-receiver time synchronization approach is typical method for most of traditional time synchronization protocols. The other hand, receiver-receiver time synchronization approach is designed in the early of 2000's. Sender-receiver time synchronization approach is worked as, the sender sends a timestamp which is contained message sending time according to the sender's clock, to receiver, then and receiver adjusts time with the sender using the received timestamp. The disadvantage of

this approach is loss of precision due to variation of message delays. Otherwise, receiver-receiver time synchronization approach reduces the influence of variation of message delays. In receiver-receiver synchronization approach, time synchronization messages are arrived at least two receivers at almost the same time, and each receiver exchanges the receiving time of the message, then each receiver estimates time offset relative to the other receiver. Obviously, this approach has advantage of the decrease of affection of the message delay variance.

2.3 Sources of Time Errors

There are various sources of time errors in the sensor networks. In this section, we address the sources of time synchronization errors and classify the sources of time errors into two types such clock drift and uncertainty of packet delays [4], [11], [12]. First, the time error in the sensor networks is caused by clock drift due to several reasons such that frequency skew, influence of temperature, and ageing. Each clock's characteristic frequency skew causes clock drift, and the characteristic frequency skew is the biggest reason of time synchronization error among the sensor nodes. In [11], sensor node's range of clock drift caused by frequency skew is $\pm$50ppm. And according to [12], physical clock of sensor has drift as $\pm$25ppm from -30C° to 70C°. Figure 2.4 illustrates the frequency versus temperature characteristics of physical AT-cut crystals oscillator.

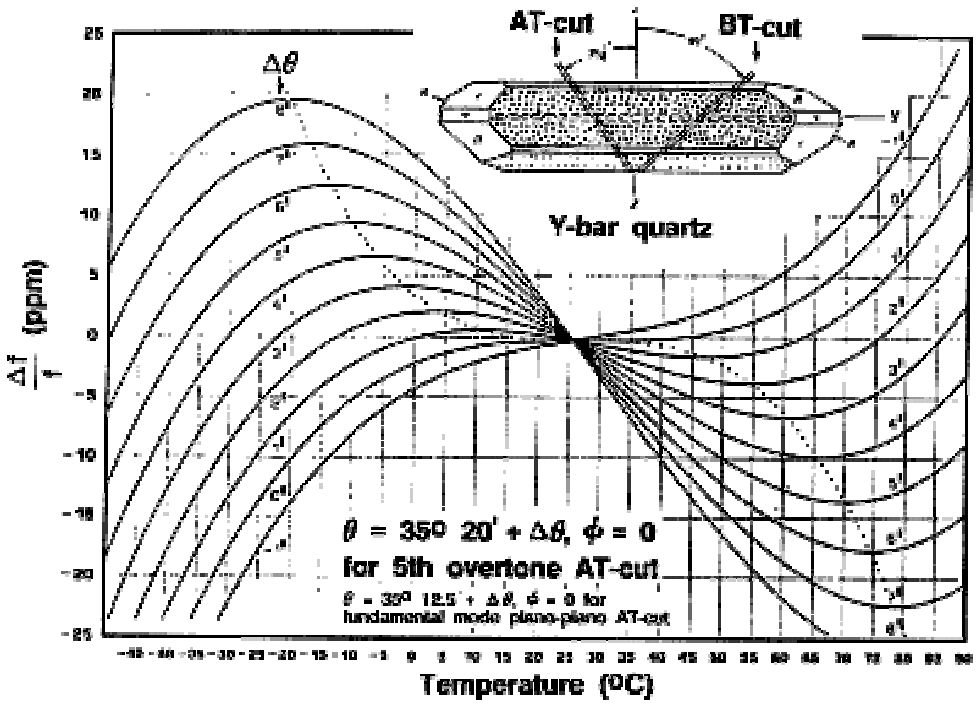

Figure 2.4 Frequency versus temperature characteristics of physical oscillator [12]

A physical sensor clock is getting slower due to ageing of clock. Ageing is defined that amount of frequency drift when operated under the specified conditions for a specified term [11]. Typically, the drift caused by ageing is only -5ppm over one year. Secondly, uncertainties of packet delay have an effect on accurate time synchronization between a transmitter and a receiver. The authors of TPSN [4] descript the uncertainties of packet delay such that send time, access time, transmission time, propagation time, reception time, and receive time. In the Table 2.2, each uncertainty factor of packet delay is explained in detail. The uncertainties of packet delay influence to accurate time adjustment between sender and receiver.

Table 2.2 Uncertainty factors of packet delay

Send time	Time delay for construction the packet at the application layer, the constructed packet takes time to reach the MAC layer from the application layer of transmitter.
Access time	After the packet reached the MAC layer, there is time delay for waiting until the packet can access the transmit channel at the transmitter side.
Transmission time	Time delay for transmitting the packet from the transmitter to the receiver through the physical layer.
Propagation time	Time delay for the packet taken time for traversing time through the wireless link from the transmitter to the receiver.
Reception time	Time delay for receiving the transmitted packet and reaching the packet to the MAC layer of the receiver side.
Receive time	Time delay for the packet reaching the application layer at the receiver.

CHAPTER 3

RELATED WORKS AND SECURITY PROBLEMS

In this chapter, we summarize existing traditional time synchronization algorithms for wireless sensor networks such that TPSN [4], RBS [1], and Tiny-Sync [8], and we claim the traditional synchronization algorithms' possible attacks in malicious circumstance. Also we present the recently published secure time synchronization, and remained secure problems.

3.1 Traditional Time Synchronization Protocols

3.1.1 Reference Broadcast Synchronization and Possible attacks

3.1.1.1 Reference Broadcast Synchronization

Reference Broadcast Synchronization (RBS) protocol [1] is based on receiver-receiver synchronization for reducing some uncertainty of packet delays such that send time, access time, transmission time, propagation time, reception time, and receive time. Caused by receiver-receiver synchronization scheme, RBS does not need to consider about sender's nondeterministic packet delays: Send time and access time. Due to remove of sender's nondeterministic packet delays, RBS provides high precision of time synchronization. Figure 3.1 shows the difference of the time critical path between sender-receiver synchronization scheme and receiver-receiver synchronization scheme.

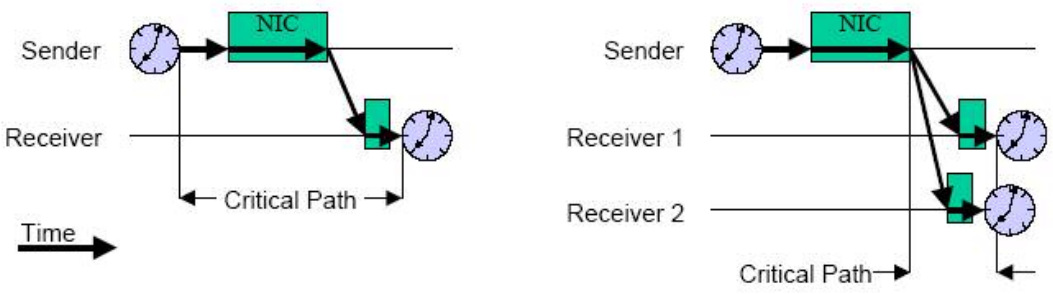

Figure 3.1 Time critical path for traditional protocol and RBS protocol [1]

In RBS protocol, at first, a beacon node broadcasts a reference packet to neighbor sensor nodes which are in the beacon node's transmission range. Each neighbor node records arrival time of the reference packet according to its own local clock. Then the reference packet received nodes exchange the recorded times each other. Finally, each receiver calculates the clock offset to any other receiver by the difference of the recorded times. For higher accuracy, the beacon node transmits multiple *m* reference packets to the receivers and each reference packet has a sequence number. After each receiver recorded *m* reference packets arrival times, the receivers exchange their observed times. Then any receiver node can compute the clock offset to any other receiver as the average of the differences for each received packet.

$$Offset[i, j] = \frac{1}{m}\sum_{k=1}^{m}(T_{j,k} - T_{i,k})$$

In above equation, i and j denote the receivers, and $T_{r,b}$ means a receiver r's recorded local time when it received a reference packet b.

3.1.1.2 Possible Attacks on Reference Broadcast Synchronization

First of all, an external adversary can overhear the beacon node's reference packet broadcasting, and then the attacker sends faulty timestamps to benign receivers during the exchange period of reference packet received times. The faultless receivers calculate the incorrect offset and clock skew with the malicious timestamps.

Second, there is jamming attack by external adversary. If the external attacker locates near the root node, the whole wireless sensor network can not make time synchronization before the adversary node stops the jamming attack. Actually, there is no way to prevent jamming attack in the wireless sensor network. However, we will not mention anymore about jamming attack when we address secure problems of other synchronization protocols.

Third, an internal compromised node can try to exchange incorrect the reference packet received times to benign neighbor receivers. This internal attack has a bad influence upon the precision of time synchronization. Moreover, only once incorrect time message exchange brings about the propagation of error throughout the whole sensor network.

Fourth, when an honest receiver has a few number of neighbor nodes and the compromise node is one of the neighbor nodes, the compromise node's disallowance of participating in the message exchange has an effect on loose estimation of clock offset and clock skew.

3.1.2 Time-sync Protocol for Sensor Networks and Possible Attacks

3.1.2.1 Time-sync Protocol for Sensor Networks

TPSN [4] is focus on providing network-wide time synchronization in establishing hierarchical structure of sensor network and pair-wise synchronization. At first, a minimum spanning tree is created at the base station which is root node. Then, TPSN works in two significance phases: The level discovery phase and the synchronization phase. The root node initiate the level discovery phase to broadcast the level discovery message with the root's level 0 to its neighbor node then the message received nodes establishes the level that is one greater than level information which is contained in the received level discovery message. After established the level, these nodes broadcast the level discovery message with their held level. Eventually, every sensor node can assign its own level. If a level assigned node received other level discovery message from other way, the node neglects and drops the message. After the end of level discovery phase, the synchronization phase is initiated by broadcasting a time-sync message at the root node. The synchronization phase is based on the sender-receiver based pair-wise synchronization. The level 1 nodes which are neighbor nodes of the root, received the time-sync message then each message received node randomly has some delay time for avoiding collision, then the node broadcasts a synchronization request packet to the root at specific time T1. The root node creates an acknowledgement which contained the receiving time of the request packet and the sending time of acknowledgement then sends to the requester the acknowledgement. When the acknowledgement is arrived to the requester at T4, the requested node

estimates its clock to the root node using obtained timestamps such that T1, T2, T3, and T4. Level 2 nodes can overhear the level 1 node's request messages because the level 2 node has at least one neighbor node which has level one. When the level 2 node overheard the level 1 node's request message, the node has some back off for random time, and then the node initiates the message exchange with the level 1 node. Finally, all of sensor nodes adjust the time with the root node through the whole network. TPSN has more accurate performance than the Reference Broadcast Synchronization algorithm. According to the simulation result in [4], TPSN shows almost twice better performance than RBS.

3.1.2.2 Possible Attack on Time-sync Protocol for Sensor Networks

First of all, during the synchronization phase, an external adversary can intercept the original time synchronization stamp from a sender, then it impersonates and transmits incorrect time stamp to a receiver node. The receiver will calculate wrong clock offset and delay with the received incorrect time stamp. Also erroneous time is propagated to the child nodes of the receiver.

Secondly, the external adversary can attack with pulse-delay attack [15]. The attacker snoops time stamp from sender and jams the signal, then it replay the snooped time stamp after some delay, Δ , to the receiver. In the receiver side, the node calculates clock offset, θ, which is affected with Δ .

Moreover, a compromise node affects to faulty time synchronization in level discovery phase and synchronization phase. TPSN requires level discovery phase before pair wise synchronization. The root node is assigned a level 0 and broadcasts a level-

discovery packet which is included sender’ s current level, i.e., level 0, to neighbor nodes. When a node received the level-discovery packet, the node assigns its level with one greater then received level. After the level discovery phase, the root node creates a minimum spanning tree of the network.

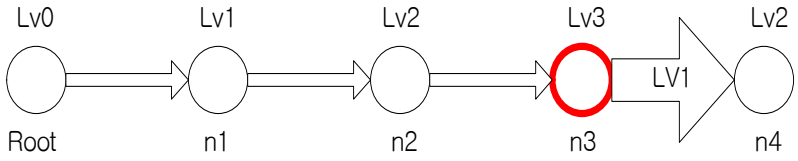

Figure 3.2 The compromise node broadcasts incorrect level

However, there are several ways the internal compromise node can attack on the level discovery phase. First, the compromise node can broadcast incorrect level to its neighbor nodes as Figure 3.2. Second, the compromise node can refuse participating in the level-discovery operation. If the compromise node is only one node between the root and the large number of child nodes like Fig, the large number of child nodes can not assign there own lever and parents. That means the nodes can not synchronize anymore in the network. Third, in the synchronization phase, the compromise node ignore request of time synchronization. Moreover, the compromise node can reply the modified timestamps to the faultless time synchronization requester.

3.1.3 Tiny-Sync and Possible Attacks

3.1.3.1 Tiny-Sync

Tiny-Sync protocol [8] is based on the set-valued identification method for time synchronization. This protocol provides deterministic bounds on the offsets and clock drifts for two relative sensor clocks with minimal computation overhead and complexity in bandwidth, storage, and processing. Tiny-Sync protocol uses two steps: The data collection step and the synchronization step. Assume that the general hardware clock's oscillator generates fixed frequency, a node i and j satisfy the following equation:

$$t_i(t) = a_{ij} t_j(t) + b_{ij},$$

Where a_{ij} and b_{ij} represent the relative drift and relative offset between two nodes i and j, and t denotes the true time. Using simple data collection algorithm, a time synchronization requester collects the three timestamps (*to, tb, tr*) for each collected data point. *to* represents the message sending time, *tb* denotes the message arriving time at the reference node. *tr* represents the ACK arriving time at the request node. Between two nodes' clocks, $a_{ij} t_b + b_{ij}$ is greater than *to* and less than *tr*. When the request node collected at least two data points, the boundaries of relative clock drift rate (a_{ij}) and relative clock offset are linearly estimated. Figure 3.3 illustrates the linear dependence of relative drift rate and offset with using collected data points.

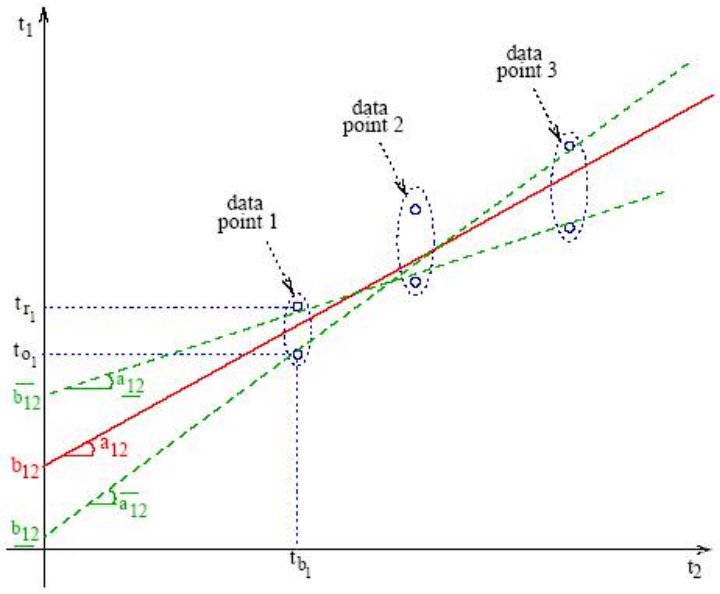

Figure 3.3 The linear dependence of the relative drift rate and offset with using data points[8]

3.1.3.2 Possible Attacks on Tiny-Sync

First of all, during the collecting time stamps, an external adversary can intercept the original time stamp (tb) from a reference node, then it impersonates and transmits incorrect time stamp (tb*) to a time stamps collector. The collector will collect wrong data point such that (to, tb*, tr), then, it will calculate wrong offset and drift with the received incorrect time stamp. Also after collection of time stamps, the receiver node estimated incorrect relative offset and drift. Second, the external adversary can attack with pulse-delay attack [13]. The attacker snoops time stamp from reference node and jams the signal, then it replay the snooped time stamp after some delay, Δ, to the collector. Therefore, in the collector side, the node obtains a data point (to, tb + Δ, tr). Also after collection of time stamps, the receiver node estimated incorrect relative offset

and drift. Third, a compromised reference node can ignore a request for time stamp from data collect node. In this case, the data collector can not gather any time stamp, it can not estimate relative offset and drift. If a compromised reference node has many neighbor nodes which did not synchronize with reference node, and the compromise node is only reference node in the deployed sensor network circumstance, the whole set of neighbors can not make time synchronization forever. Fourth, if the compromise reference node modifies the time stamp, and sends to data collect node, the data collect node obtains a incorrect data point such that (to, tb', tr). After collection of time stamps, the receiver node estimated incorrect relative offset and drift. Also the incorrect time will be propagated to other sensor nodes.

3.2 Secure Time Synchronization Algorithms and Remained Problems

3.2.1 Secure Pairwise Synchronization

Secure Pairwise Synchronization (SPS) protocol [14] is designed against pulse delay attack by external attacker. Pulse delay attack is defined that the external or internal attacker intentionally delays some of the messages which involved time synchronization. In SPS, integrity and authentication are secured through the Massage Authentication Codes (MAC) and the shared secret key between two related sensor nodes. Also the use of random nonce ensures the time message's freshness against replay attack. SPS protocol is designed on sender-receiver based pair wise synchronization like as TPSN. And using the simulation result of end-to-end delay over a single link, each sensor stored the maximum end-to-end delay before the sensors are deployed. When the sensors are in time synchronization operation, the sensors detect

external attacker's delay attack through a comparison of the maximum delay and established delay. If the established delay is greater than the maximum end-to-end delay, the sensor aborts the offset calculation and time adjustment. However, SPS protocol can not resilient to malicious attacks from internal attackers. If one or more sensors are compromised and these sensors broadcast incorrect timestamps to innocent sensor nodes, the faultless sensors synchronize those clocks with erroneous timestamps.

3.2.2 Attack-Resilient Time Synchronization

Song, Zhu, and Cao's Attack-Resilient Time Synchronization protocol [13] is based on receiver-receiver synchronization like as RBS. This secure time synchronization protocol uses the generalized extreme studentized deviate (GESD) algorithm against malicious node's delay attack. GESD algorithm is an improved version of the Extreme Studentized Deviate Test which is good at detecting one outlier which is defined as "an observation which deviates so much from other observations as to arouse suspicious that it was generated by a different mechanism" [9]. GESD can detect multiple outliers, which are delay attackers, using the set of the time offsets and the median of the time offset set, and standard deviation. However, GESD needs a large number of time offsets for detecting the malicious attackers. It is obviously drawback due to resource consumption for gathering many timestamps and computing the standard deviation.

CHAPTER 4

RELATIVE CLOCK DRIFT RATE BASED SECURE TIME SYNCHRONIZATION

In previous chapter, we studied existing time synchronization algorithms' secure holes caused by external or internal malicious nodes. In this chapter, we propose Relative clock drift rate based Secure Time Synchronization (RSTS) protocol which is countermeasure for malicious attacks by compromise nodes in the wireless sensor networks. First, we address how to estimate the maximum and the minimum relative clock drift rate between sender and receiver sensor with considering expected two types of sources of time error: Clock drift and uncertainty of packet delay. And then we determine thresholds for boundary of acceptable offset for next synchronization phase between the relative two sensors with the use of the estimated maximum and the minimum relative clock drift rates.

4.1 Assumptions

Every sensor node has an own physical clock which is assisted by an oscillator, and all physical clocks have peculiarity clock drifts. And every sensor is working on limited external battery. We assume that all of deployed sensors have unique identifiers, and each node shares a secret key with its neighbor node for Message Authentication Code (MAC) for providing integrity and authentication, and uses a random nonce for freshness of timestamps against the replay attack. Figure 4.1 shows how to use the

shared secret key, MAC and random nonce over the sender-receiver based pair wise time synchronization.

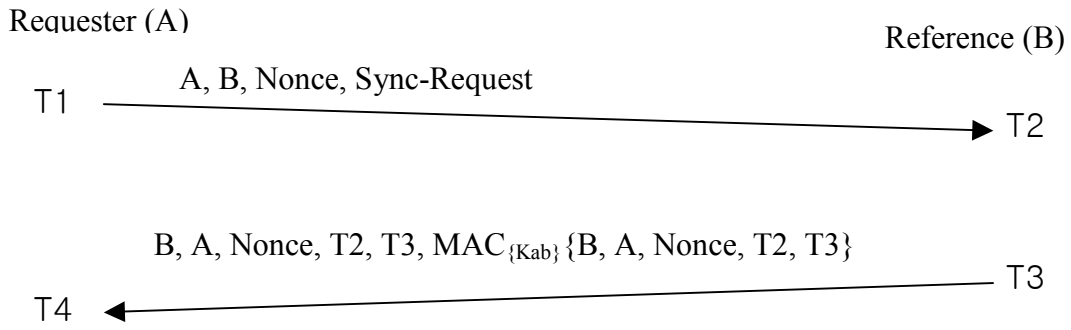

Figure 4.1 Sender-receiver based pair wise time synchronization with using shared secret key, MAC, and random nonce.

And the network use a sender-receiver based pair wise time synchronization like as TPSN. When a sensor broadcasts time message to other sensors which are in the sensor's radio transmission range, each message takes over in bounds of the maximum and the minimum delay times. All nodes initiate at same time, and the nodes have a consistent frequency skew for each sensor's physical clock. Periodically, each node has to re-synchronize after specific interval. And the sensor nodes are in a secure circumstance at least a short time period without any compromise node's attacks after these are initially deployed, because an adversary needs some time for capturing the sensors and programming malicious codes in the captured sensors [5]. Therefore, we assume that all of the deployed sensors synchronize at least one time without any malicious attack.

4.2 Estimation of Relative Clock Drift Rate

In this section, we will propose simple and novel method for estimating the maximum and the minimum relative clock drift rate between two nodes by using calculated offset and synchronization interval in sender-receiver based pair wise time synchronization with considering of expected time errors.

In sender-receiver based pair wise synchronization scheme, node A which is requester of time synchronization, sends time synchronization request packet to node B which is reference node, at time T1. And B receives A's request packet at T2, then it sends ACK which includes timestamps T2 and T3, to A. After A received ACK from B at T4, A can calculates an offset relative to the reference node B.

$$\theta_C = \frac{(T2 - T1) - (T4 - T3)}{2},$$

where θ_C expresses calculated offset.

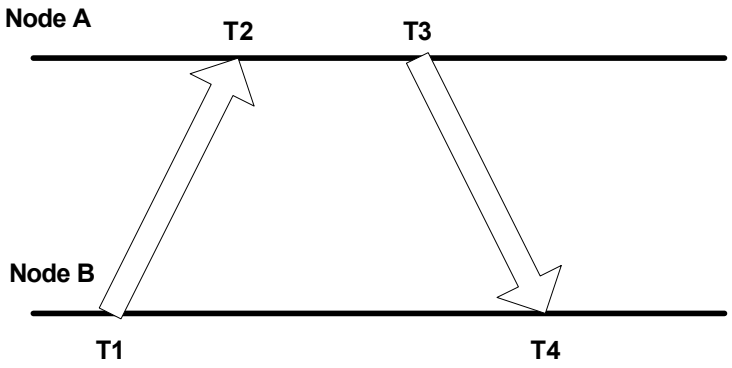

Figure 4.2 Message exchange in the sender-receiver based pair wise time synchronization

According to our assumptions, node A and B initiated at same moments. So the calculated offset is occurs for interval from the time of initiation to ACK receiving time (T4) at first time synchronization phase in the network. Than, we have to consider about the uncertainty of packet delay, because, referred to TPSN [4], the sender-receiver based pair wise time synchronization algorithm has a time error Δ due to uncertainty of packet delays such that send time, access time, transmission time, propagation time, reception time, and receive time. Therefore we can derive,

$$\theta_R = \theta_C + \Delta \quad ....(1)$$

where θ_R denotes the real offset between two nodes.

When B reports time T4, A's local time is expressed T4 * (1/ Relative Drift Rate) with our assumption about initiation time. And then we can derive equation (2) for representing the real relative offset with above factors,

$$\theta_R = [(T4 - Ti) \times \frac{1}{\alpha_R^{B \to A}}] - (T4 - Ti) \quad ...(2)$$

where $\alpha_R^{B \to A}$ denotes B's real drift rate relative to node A, *Ti* indicates the initiation time for all nodes.

And, we can derive equation (3) with the uses of (1) and (2).

$$\theta_C + \Delta = [(T4 - Ti) \times \frac{1}{\alpha_R^{B \to A}}] - (T4 - Ti) \quad ...(3)$$

If we divide (*T4-Ti*) to each side in (3),

$$\frac{\theta_C + \Delta}{(T4 - Ti)} = \frac{1}{\alpha_R^{B \to A}} - 1$$

$$\frac{\theta_C + \Delta}{(T4 - Ti)} = \frac{1}{\alpha_R^{B \to A}} - 1 = -(1 - \frac{1}{\alpha_R^{B \to A}})$$

Since $\alpha_R^{B \to A}$ is equal to $1+(1-\frac{1}{\alpha_R^{B \to A}})$, we can draw the equation (4) for the real relative clock drift rate.

$$1 - \frac{\theta_C + \Delta}{(T4 - Ti)} = 1 + (1 - \frac{1}{\alpha_R^{B \to A}}) = \alpha_R^{B \to A}$$

$$1 - \frac{\theta_C + \Delta}{(T4 - Ti)} = \alpha_R^{B \to A} \quad ...(4)$$

Time error Δ caused by uncertainty of packet delay is actually uncalculating factor. But, we can derive the maximum and minimum possible time errors with system conditions such as the measured maximum and minimum packet delays, and the maximum clock drift rate given by manufacturer. If we estimate the maximum and minimum possible time errors, we can obtain the range of the maximum and minimum relative clock drift rate with equation (4).

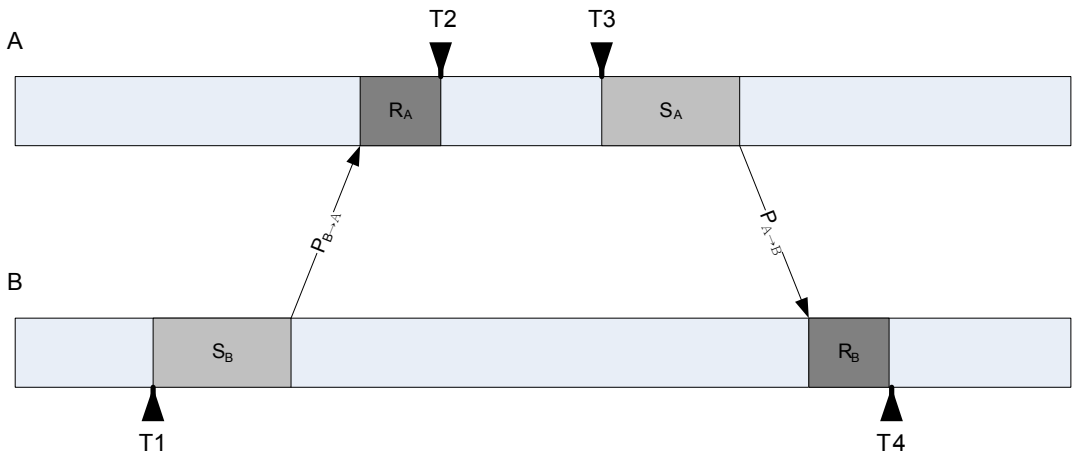

Figure 4.3 Analysis of uncertainty of packet delay

Figure 4.3 shows the uncertainty of packet delay in sender-receiver based pair wise time synchronization. According to time error analysis in [4], S_B denote to the summation of send time, access time, and transmission time at node B. $P_{B\rightarrow A}$ refer to propagation time from B to A, and R_A express the summation of reception time and receive time at node A. The author of TPSN [4] expressed the time error Δ like as,

$$\text{Time error } \Delta = \frac{(S_B + P_{B\rightarrow A} + R_A) - (S_A + P_{A\rightarrow B} + R_B)}{2} + \frac{RD_{t1\rightarrow t4}^{B\rightarrow A}}{2},$$

where $RD_{t1\rightarrow t4}^{B\rightarrow A}$ denotes to the relative drift between the node A and B from the true time t1 to t4. The difference of A's drift and B's drift during the interval [t1, t4] can estimate with the real relative clock drift between two nodes. So we can express to the relative drift between the node A and B by the interval [t1, t4], and the drift rates of node A and B:

$$RD_{t1\rightarrow t4}^{B\rightarrow A} = [(t4 \times DR^B) - (t1 \times DR^B)] - [(t4 \times DR^A) - (t1 \times DR^A)] \quad ...(5),$$

where DR^A denotes A's drift rate relative to real time, and DR^B denotes B's drift rate relative to real time. And, in equation (5), the true times and the drift rate in each parentheses can be replaced the known time factors, T1 and T4, and $\alpha_R^{B\rightarrow A}$ like as (6).

$$t4 \times DR^B = T4, \qquad t1 \times DR^B = T1,$$

$$t4 \times DR^A = T4 \times \frac{1}{\alpha_R^{B\rightarrow A}}, \qquad t1 \times DR^A = T1 \times \frac{1}{\alpha_R^{B\rightarrow A}},$$

therefore,

$$RD_{t1\to t4}^{B\to A} = (T4-T1) - [(T4\times\frac{1}{\alpha_R^{B\to A}}) - (T1\times\frac{1}{\alpha_R^{B\to A}})]$$

$$= (T4-T1) - (T4-T1)\times\frac{1}{\alpha_R^{B\to A}} \qquad .....(6)$$

When we replace ($S_B + P_{B\to A} + R_A$) and ($S_A + P_{A\to B} + R_B$) to Delay1 and Delay2,

$$Delay1 = S_B + P_{B\to A} + R_A$$

$$Delay2 = S_A + P_{A\to B} + R_B$$

$$\Delta = \frac{Delay1 - Delay2}{2} + \frac{(T4-T1) - (T4-T1)\times\frac{1}{\alpha_R^{B\to A}}}{2},$$

However, Δ is uncertain value in the sender-receiver based pair wise time synchronization. Also each node can not know about actual packet delay time and the real relative clock drift rate between two nodes. But according to our assumption, the packet delays are in boundary of maximum and minimum delay. And using given clock drift information by manufacturer, we can obtain maximum and minimum possible time errors caused by uncertainty of packet delay for sender-receiver based pair wise time synchronization.

$$\Delta_{\max} = \frac{MaxDelay - MinDelay}{2} + \frac{(T4-T1) - (T4-T1)\times\frac{1}{\alpha_{\max}}}{2} ....(7)$$

$$\Delta_{\min} = \frac{MinDelay - MaxDelay}{2} + \frac{(T4-T1) - (T4-T1)\times\alpha_{\max}}{2} ....(8)$$

where Δ_{max} and Δ_{min} represent the obtain maximum and minimum possible time errors caused by uncertainty of packet delay, and α_{max} denotes the given maximum clock drift information by manufacturer.

Finally, we can derive the equation (9) with using equation (7), (8), and (4),

$$1 - \frac{\theta_C + \Delta_{max}}{(T4 - Ti)} \leq \alpha_R^{B \to A} \leq 1 - \frac{\theta_C + \Delta_{min}}{(T4 - Ti)} \quad ...(7)$$

$$\alpha_{E\_min}^{B \to A} \leq \alpha_R^{B \to A} \leq \alpha_{E\_max}^{B \to A}$$

where $\alpha_{E\_min}^{B \to A}$,and $\alpha_{E\_max}^{B \to A}$ denote minimum and maximum estimated relative clock drift rates.

With the equation (9), we simulated the accuracy of minimum and maximum estimated relative clock drift rates. In the computational simulation, we set one node has 30ppm of clock drift and the other node has -10ppm of clock drift. In this case, the real relative clock drift rate $\alpha_R^{B \to A}$ is 0.99996. For maximum and minimum delay, we referred [14]'s examination, boundary of delay [753.54us, 770.46us]. For the maximum and minimum clock drift information given by manufacturer, we assumed all physical clocks have clock drift [10ppm, 50ppm]. And two sensor nodes will have random delay between maximum and minimum delays. Figure 4.4 shows the result of computational simulation. According to the result, longer re-synchronization interval guarantees estimation of tighter the maximum and the minimum relative clock drift rate.

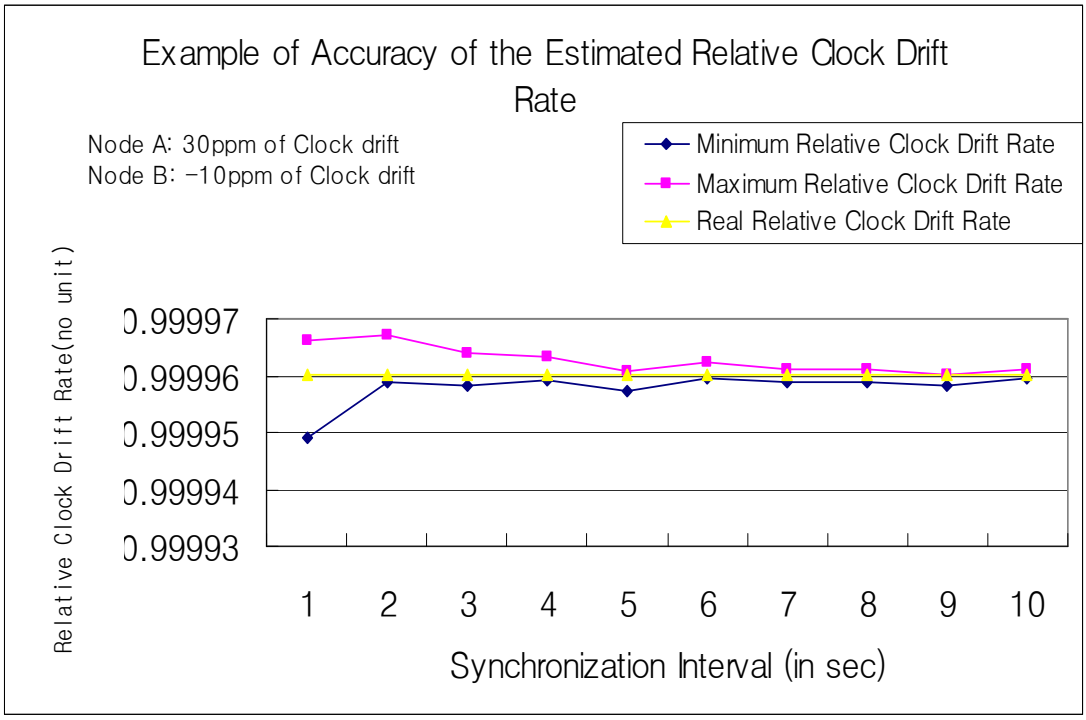

Figure 4.4 Computational result of accuracy of the estimated maximum and the minimum relative clock drift rate between two nodes

4.3 Determination of Thresholds for Acceptable Offset

In the previous section, we addressed how to estimate the relative clock drift rate between sender node and receiver node. In this section we present tight bounds for acceptable offset in a malicious circumstance.

The sensor node has own time error by the characteristic clock drift as time passed after initial time synchronization through whole sensor network, even there is not any malicious attack. Since increase of time error, the sensor network requires the re-synchronization process. From second synchronization process, we have to consider about the adversary's attack especially the compromise nodes' incorrect timestamps propagations. This type of malicious attack can not be solved by existing time

synchronization algorithms such as TPSN, RBS, Tiny-Sync, and SPS. However, the uses of the estimated maximum and minimum relative clock drift rate provides the bounds of acceptable offset for next time synchronization process after specific time period. In the initial time synchronization process, each sensor node estimated the maximum and the minimum clock drift rate relative to its reference node.

If the reference nodes are non-faulty, each sensor node calculates the offset which is in the boundary of the maximum expected offset and the minimum expected offset according to the estimated clock drift rates relative to the reference node. When node A is the reference node and B is the time synchronization requester, B estimated the maximum and the minimum clock drift rate relative to the reference node A in the initial time synchronization process. After some interval, the request node has to re-adjust the clock to the reference node due to the increase of time error. Therefore B broadcasts the time request message to A, and A replies ACK to B. And then node B can calculate the offset using the collected timestamps. At this time, node B must check the calculated offset is greater than $(\frac{1}{\alpha_{E\_\max}^{B\to A}}-1)\times Interval$, and less than $(\frac{1}{\alpha_{E\_\min}^{B\to A}}-1)\times Interval$, where *Interval* denotes the re-synchronization period. Easily we can defines the *Interval* is the time period from the latest time synchronization point to the ACK arriving time point (T4) in this synchronization process.

$$\theta_{\max} = (Interval \times \frac{1}{\alpha_{E\_\min}^{B\to A}}) - Interval = (\frac{1}{\alpha_{E\_\min}^{B\to A}} - 1) \times Interval$$

$$\theta_{\min} = (Interval \times \frac{1}{\alpha_{E\_\max}^{B\to A}}) - Interval = (\frac{1}{\alpha_{E\_\max}^{B\to A}} - 1) \times Interval$$

$$(\frac{1}{\alpha_{E\_\max}^{B\to A}} - 1) \times Interval \le Offset \le (\frac{1}{\alpha_{E\_\min}^{B\to A}} - 1) \times Interval$$

If the calculated offset is not in the bounds of the acceptable offset, the time synchronization requester detects that the reference node is compromised and aborts the synchronization processing for adjustment to the reference clock.

4.4 Updating the Estimated Maximum and Minimum Relative Clock Drift Rates

After successful synchronization, the requester has to updates the maximum and minimum relative clock drift rates because each sensor's drift rate can be changed due to influence of environment factors such as temperature, pressure, and battery power. In this thesis, we use the forgetting factor [31] for updating the estimated maximum and minimum relative clock drift rates with given more weight to the latest estimated relative clock drift rates. Typically, forgetting factor λ value is 0.95 to 1 [31]. Where N denotes number of synchronization period, $\overline{\alpha}_{\min\_N}^{B\to A}$ denotes Nth updated minimum relative clock drift rate and $\overline{\alpha}_{\max\_N}^{B\to A}$ expresses Nth updated maximum relative clock drift rate, the estimated can be updated with the uses of following equations.

$$\overline{\alpha}_{E\_\min\_N}^{B\to A} = \alpha_{E\_\min\_N}^{B\to A} \qquad \text{if } N = 1$$

$$\overline{\alpha}_{E\_\min\_N}^{B\to A} = \lambda \cdot \alpha_{E\_\min\_N}^{B\to A} + (1-\lambda) \cdot \overline{\alpha}_{E\_\min\_(N-1)}^{B\to A} \qquad \text{if } N > 1$$

$$\overline{\alpha}^{B\to A}_{E\_\max\_N} = \alpha^{B\to A}_{E\_\max\_N} \qquad \text{if } N = 1$$

$$\overline{\alpha}^{B\to A}_{E\_\max\_N} = \lambda \cdot \alpha^{B\to A}_{E\_\max\_N} + (1-\lambda)\cdot \overline{\alpha}^{B\to A}_{E\_\max\_(N-1)} \qquad \text{if } N > 1$$

Table 4.1 Relative clock drift rate based secure time synchronization

Relative Clock Drift Rate Based Secure Time Synchronization Protocol

First Time Synchronization phase

1. B requests time stamps to A at T1
2. A receives request packet from B at T2
3. A prepares ACK packet, which includes T1, T2, T3, and sends ACK to B
4. B receives ACK at T4, then,

$$\theta_C = \frac{(T2-T1)-(T4-T3)}{2}$$

Time Synchronization with the use of θ_C

LastSyncPoint = Synchronized current time

$$\Delta_{\max} = \frac{MaxDelay - MinDelay}{2} + \frac{(T4-T1)-(T4-T1)\times\frac{1}{\alpha^{B\to A}_{\max}}}{2},\quad \Delta_{\min} = \frac{MinDelay - MaxDelay}{2} + \frac{(T4-T1)-(T4-T1)\times\alpha^{B\to A}_{\max}}{2}$$

$$\alpha^{B\to A}_{E\_\min} = 1 - \frac{\theta_C + \Delta_{\max}}{(T4 - Ti)},\quad \alpha^{B\to A}_{E\_\max} = 1 - \frac{\theta_C + \Delta_{\min}}{(T4-Ti)}$$

Second Time synchronization phase

1. B requests time stamps to A at T1
2. A receives request packet from B at T2
3. A prepares ACK packet, which includes T1, T2, T3, and sends ACK to B
4. B receives ACK at T4, then,

If (T1 ≠ ACK.T1) **then** Abort current time synchronization phase

$$\theta_C = \frac{(T2-T1)-(T4-T3)}{2}$$

If $\left((\frac{1}{\alpha^{B\to A}_{E\_\max}} - 1)\times(T4 - LastSyncPoint) < \theta_C < (\frac{1}{\alpha^{B\to A}_{E\_\min}} - 1)\times(T4 - LastSyncPoint)\right)$ **then** {

Time Synchronization with the use of θ_C

Update the estimated maximum and minimum relative clock drift rates

}

Else

{

Abort current time synchronization phase

LastSyncPoint = Synchronized current time

}

CHAPTER 5

EXPERIMENTAL IMPLEMENTATION

In previous chapter, we studied RSTS protocol which is countermeasure for malicious attacks by compromised internal nodes in the wireless sensor networks. In this chapter, we describe the environment of experimental implementation, and then we show the results of implementation with the uses of MICA2 motes for evaluation of performance of RSTS.

5.1 Implementation Environment

We have implemented RSTS for the MICA2 sensor motes which are 3rd generation of wireless smart sensors [24] and developed by University of California in Berkeley [25]. The sensor motes use ATmega Atmel 128 microprocessor with 128K bytes program flash memory, 4K bytes configuration EEPROM, and 512K bytes measurement flash. 50 multiple radio channel with 916 MHz bandwidth is supported for communication between MICA2 motes [24]. MIB510CA mote interface board is used a base station and the board provides a serial interface for programming and communication with PC which uses Window XP for OS. We implemented RSTS in nesC programming language [29] for TinyOS version 1.1.15 [27]. For RSTS, we modified the published TPSN application [28], [30]. In existing TPSN application we removed level set up operation and added secure functions such as estimation of maximum and minimum relative clock drift rates, and determination of thresholds. Also

we made up for communication function with TOS-Base Station (MIB510CA board) for monitoring synchronization and detection. When TOS-Base Station received reporting packet from MICA2 motes, the TOS-Base Station sends the received packet to PC through the serial cable. Then a monitoring application which is developed by JAVA, reads and displays the reporting message on the PC's monitor.

In previous chapter 4, we assumed that every sensor is initiated at same time. But this assumption is impossible when we implemented with real hardware sensor. For this reason, each sensor adjusts clock at first synchronization phase, than the maximum and minimum relative clock drift rates are estimated at second synchronization phase.

In this implementation, we did not provide the basic secure mechanisms such as shared secret key, Message Authentication Code (MAC) and random nonce.

For evaluation of detecting performance, we set one sensor, the time reference node, acts compromise node. Periodically the time synchronization requester broadcasts the request packet. When the compromised time reference node received the request packet from the other node as requester, the reference node sends ACK packet which is included timestamps (T2 and T3). Once in three times, the compromise node sends modified timestamps with adding particular time; 100 microseconds, 1 millisecond, and 10 milliseconds. After the requester received timestamps, the node calculate offset with the uses of the timestamps then determines whether timestamps are attacked or not by using estimated maximum and minimum relative clock drift rates and synchronization interval which is subtraction a last synchronized point (Ti) from the ACK packet receiving time (T4). If the requester mote detects the modified timestamps, the node

aborts current time's synchronization. The other hand, the calculated offset is in the range of the estimated thresholds, the requester adjustment the clock using the calculated offset. Finally, the requester sensor provides reporting packet with result of detection, the calculated offset, and the estimated thresholds. The monitoring application runs until it collects 100 reporting messages.

For measuring the accurate time with microsecond unit, RSTS application uses AVR Timer3 [30] like as TPSN application and default timer in TinyOS. The default timer supports only minimal time unit with millisecond for time scale in TinyOS version 1.x. Timer3 addresses the time with *TCNT3* which is Timer3 counter with 16 bits instead common time units. If *TCNT3*'s register is full, the timer increases one to *MTicks* variable. Since, there is not any reference one *TCNT3* denotes how much common time such as microsecond, we used both of the default timer and Timer3 for measuring. When a sensor mote was initiated, two timers started almost same time, then, the default timer fired after 100 seconds and Timer3 reported its time by *MTicks* and *TCNT3*. According to our measurement, Timer3 reported 173 *Mticks* and 65100 *TCNT3* during 100 seconds of the default time. Therefore, one *TCNT3* denotes 8.7698 microseconds.

5.2 Evaluation

For estimation of maximum delay and minimum delay, we ran the RSTS on 3 different pairs of MICA2 motes, and we collected 100 delay samples for each pair of motes. Figure 5.1 illustrates measured delays for 100 iterations for each pair of MICA2 motes and Table 5.1 shows the statistics about 300 measured delay samples.

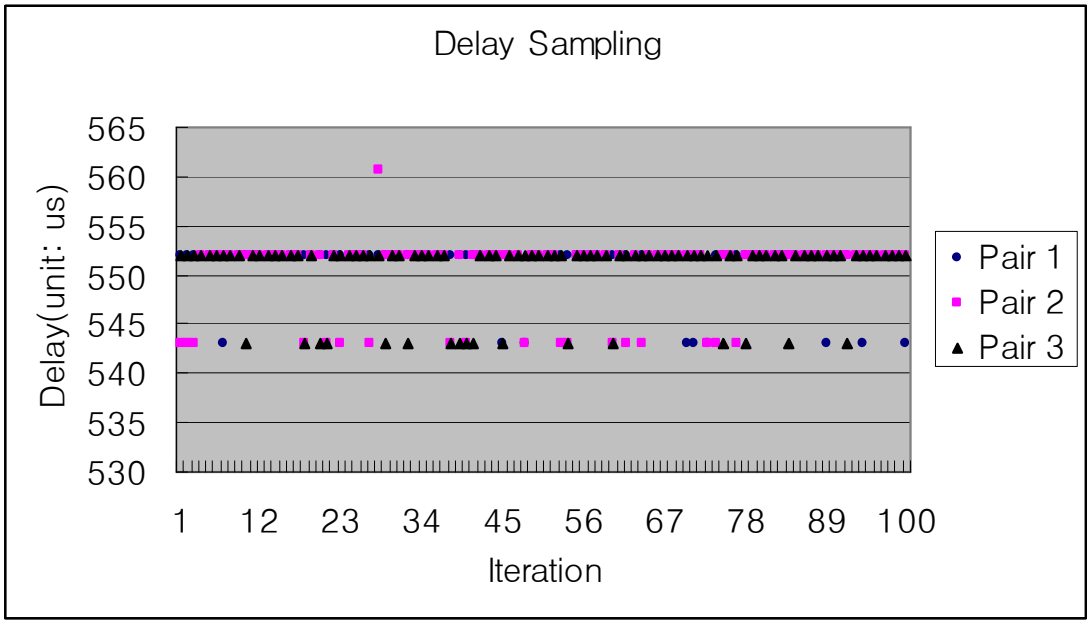

Figure 5.1 Delay samples

Table 5.1 Statistics of delay samples

# of Samples	Max delay	Min delay	Average	Std dev
300	560.64 us	543.12 us	550.6244 us	3.157108 us

In our implementation for RSTS, we assumed maximum clock drift which is given by manufacturer, is ±100ppm. Therefore, the maximum clock drift rate is 1.0001 and the minimum clock drift rate is 0.9999. That means the maximum relative between any two sensors is 1.0002 and the minimum relative clock drift rate is 0.9998. Using these determined factors, *MaxDelay* and *MinDelay*, we ran RSTS application. Figure 5.2 illustrates the result of estimation of relative maximum and minimum clock drift rates under variable synchronization interval. According to this result, longer interval affect to tighter estimation of relative clock drift rates.

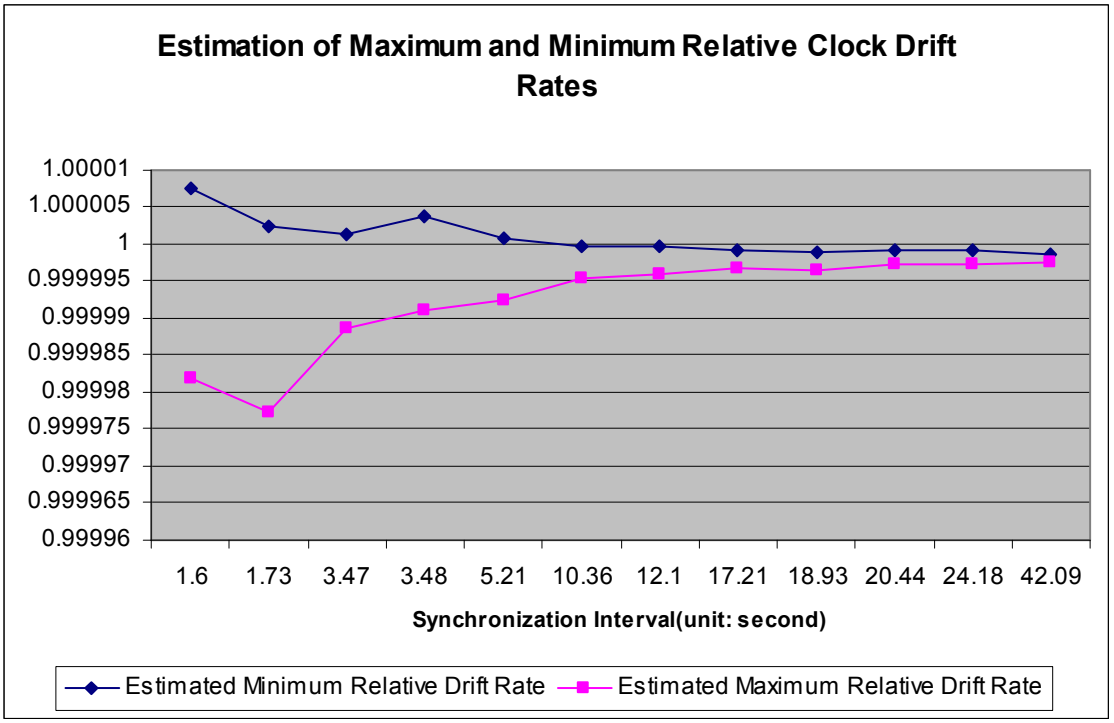

Figure 5.2 Estimation of maximum and minimum relative clock drift rates between sender and receiver.

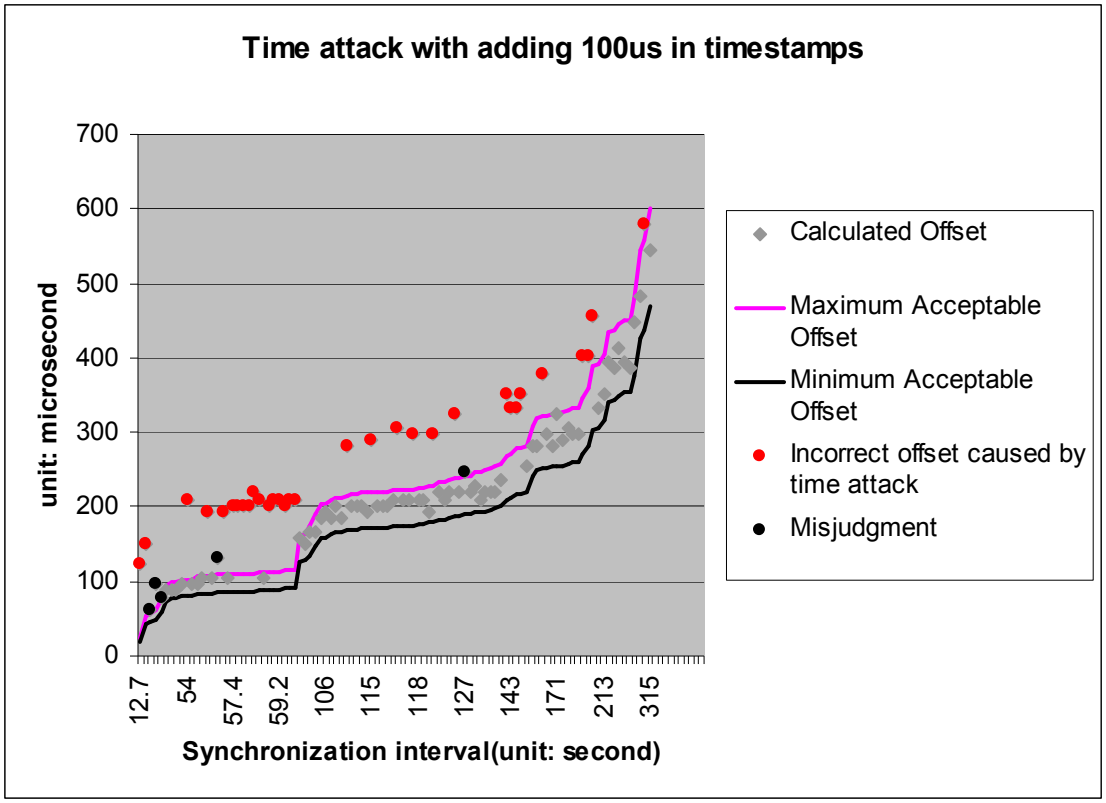

Figure 5.3 RSTS runs 100 times with possibility of time attack with adding 100us in timestamps

Figure 5.3 shows the 100 times of runs with RSTS under possibility of time attack with adding 100us in timestamps such as T2 andT3. The compromised reference sensor attacks 32 times out of 100 synchronization phases. And the time requester sensor successfully detects all of attacks with 100% of detection rate. But, the requester misjudges 5 times in non-faulty situations (5% of misjudgment rate). When we ran RSTS under short synchronization interval with possibility of time attack with adding 100us in timestamps, we obtained worse result than longer synchronization interval circumstance. Figure 5.4 describes the result of the 100 times of runs with RSTS under short interval. In this result, there are 9 times of failed detection, and 8 times

misjudgments. For reference with Figure 5.5 and Figure 5.6, RSTS achieved a 100% successful detection performance when the compromise node attacked with adding over 1ms. And Figure 5.7 shows implementation result of multi-hop secure synchronization with three Mica2 sensor motes. Node B is placed in the middle of root node A and end node C, and B synchronize with node A. And B can be the reference node for the end node C. In this implementation, B attacks to the end node with adding 1 ms. Figure 5.8 illustrates the detection rate under various synchronization intervals. According to this result, when the synchronization interval is over 2 minutes, RSTS performs well against multiple attacks even attack time is 100us.

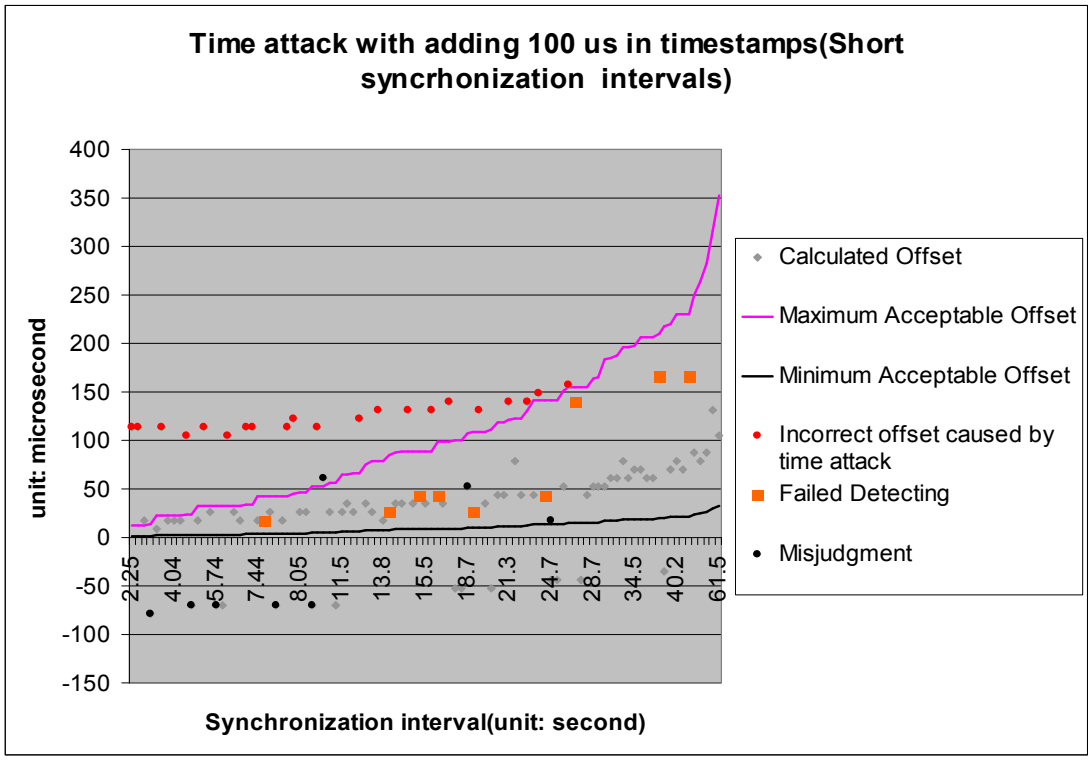

Figure 5.4 RSTS runs 100 times with possibility of time attack with adding 100us in timestamps under short synchronization intervals

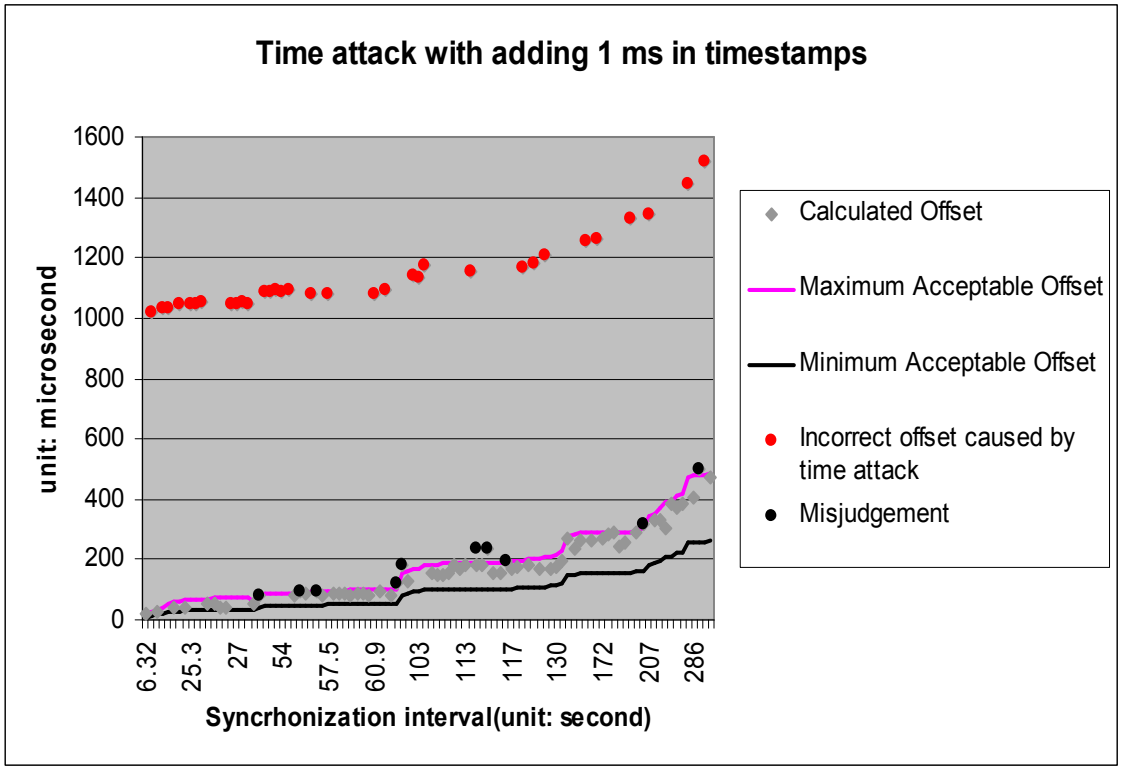

Figure 5.5 RSTS runs 100 times with possibility of time attack with adding 1ms in timestamps

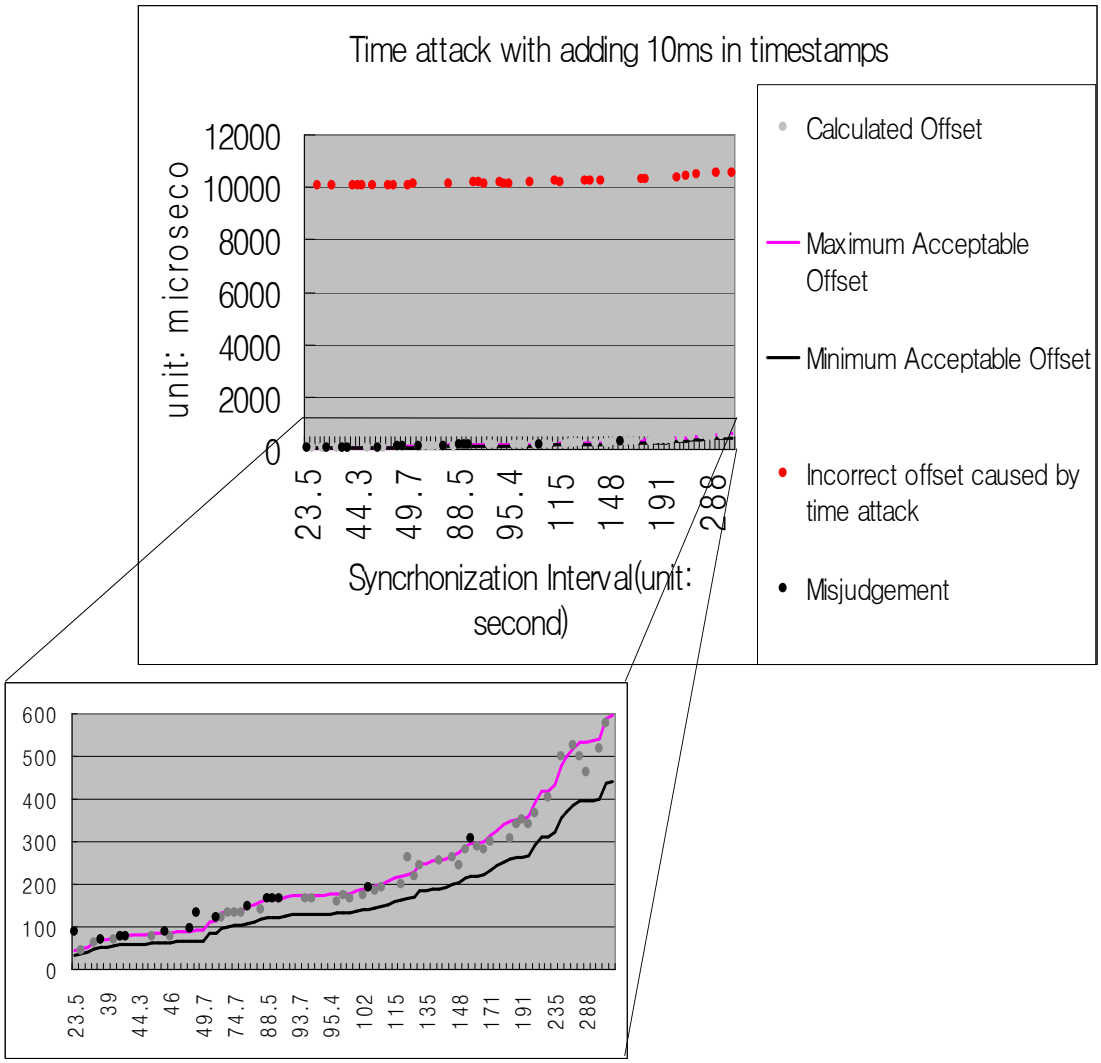

Figure 5.6 RSTS runs 100 times with possibility of time attack with adding 10ms in timestamps

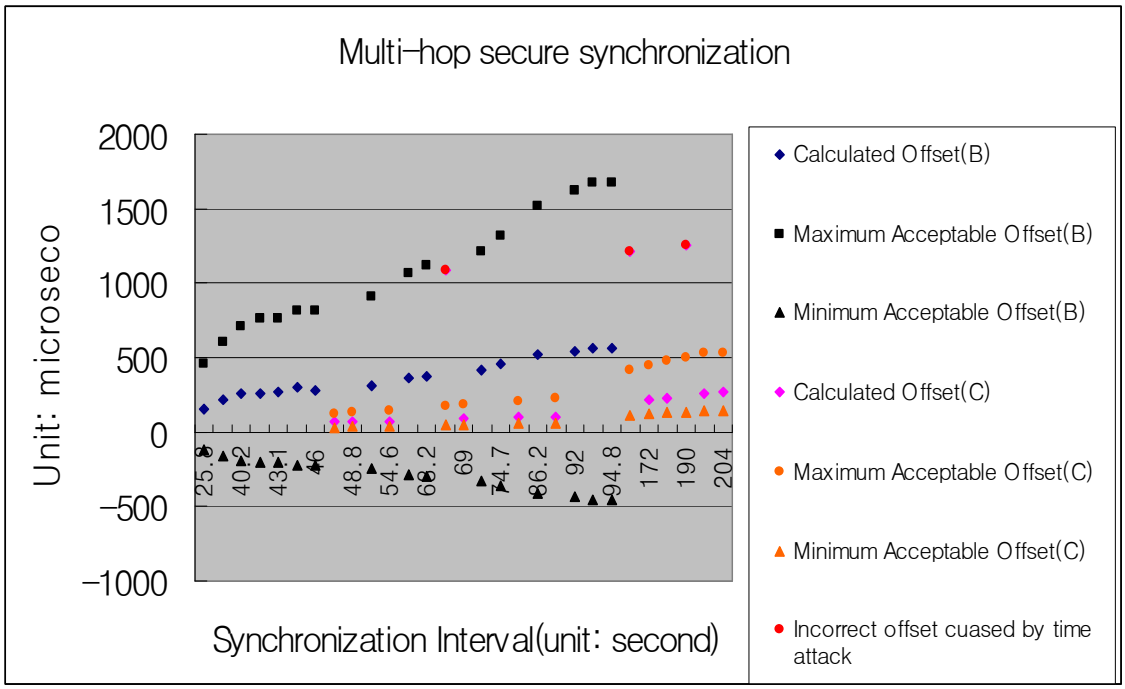

Figure 5.7 RSTS runs 30 times with 3 tiers of sensor under possibility of time attack with adding 1ms in timestamps

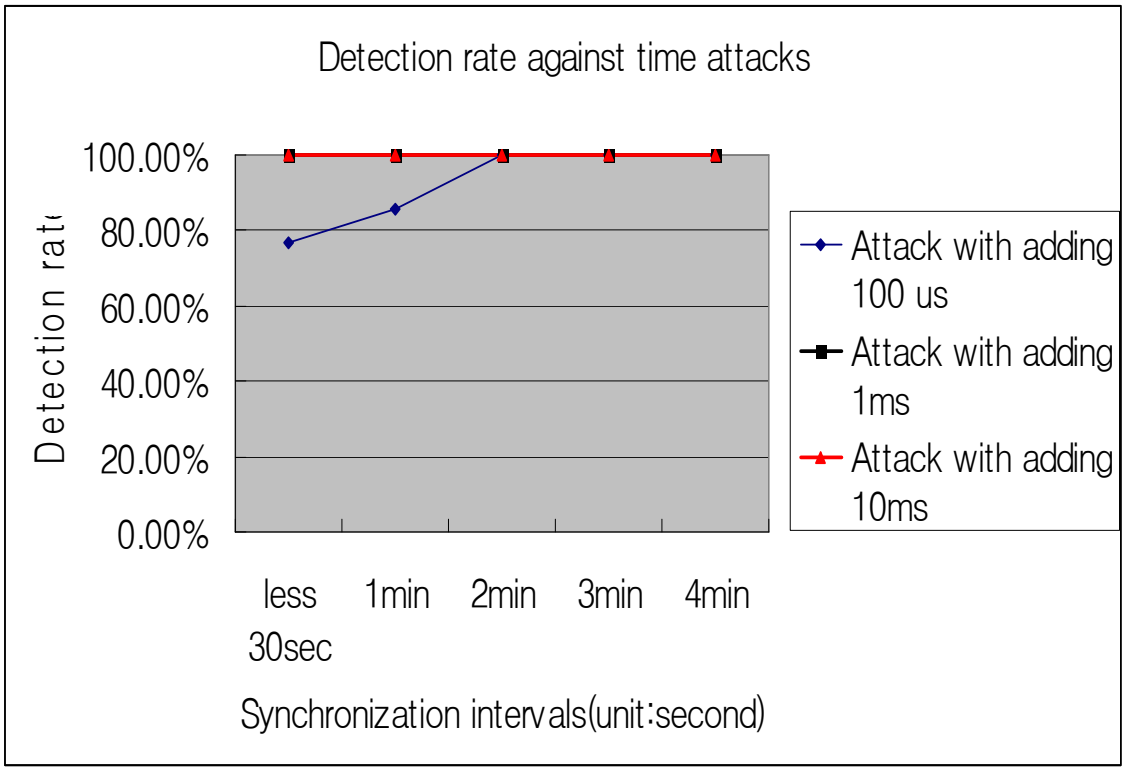

Figure 5.8 Detection rate against multiple time attack under various synchronization intervals

In Figure 5.9, we summarize the result of misjudgment rate under multiple synchronization intervals. Within 432 non-attack synchronizations, RSTS faultily claims total 49 times. That means our RSTS implementation shows 11.34% of misjudgments rate. In detail, when a synchronization interval is longer than 4minutes, RSTS shows higher misjudgment rate. The other hand, when synchronization intervals are 3 to 4 minutes, RSTS shows more accurate rate of correct determinations. With analyzing the misjudged cases, 49% of cases have under 8.77us (equal to one *TCNT3*) of offset error.

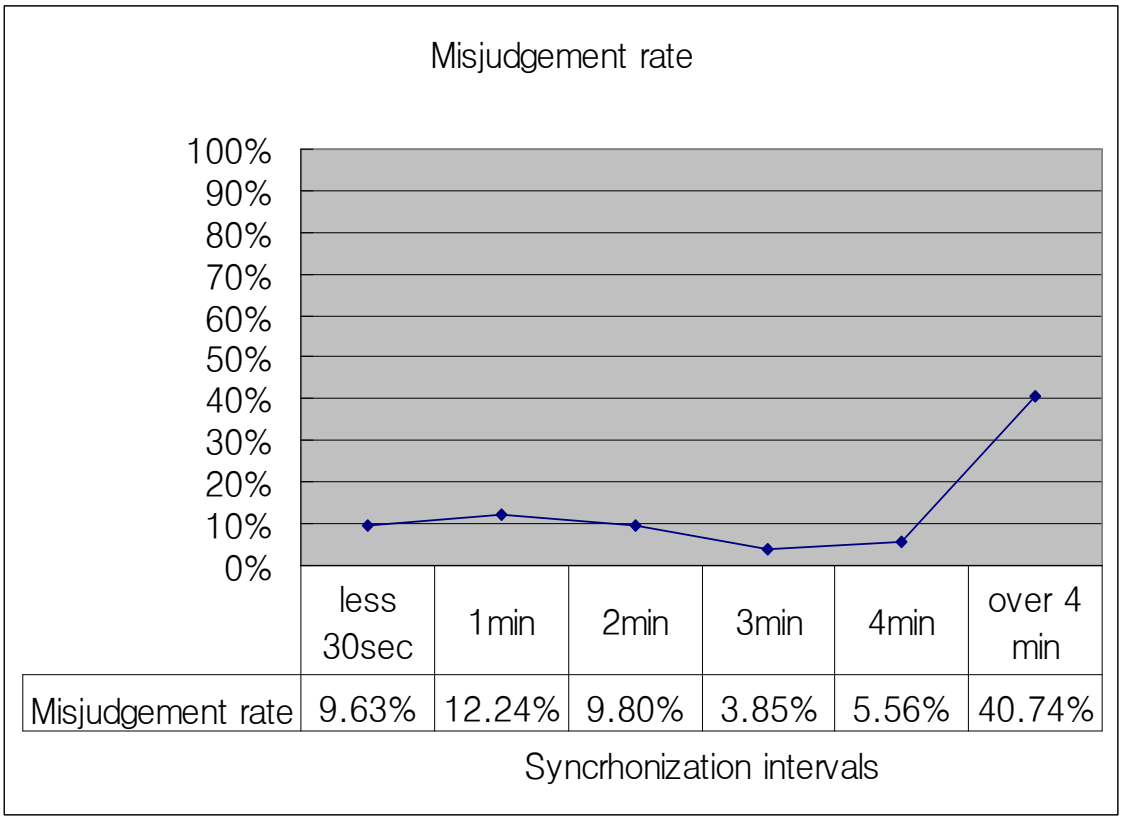

Figure 5.9 Within 432 non-attack synchronization phases, RSTS misjudged 49 times.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

Time synchronization is one of self-reliant part of foundation for wireless sensor networks. For solid infrastructure in sensor network, time synchronization should be guarded by secure function. For secure time synchronization, we have introduced novel mechanism, RSTS, which is based on the sender-receiver synchronization. All of existing secure time synchronization algorithms based on the sender-receiver synchronization do not protect a sensor network from compromise nodes' attack with modification of reference timestamps. For detecting of malicious time attack, RSTS does require minimal computational overhead. Moreover, RSTS employs the same consumption of bandwidth as other existing un-secure time synchronization algorithms. RSTS provides a simple estimation method of maximum and minimum relative clock drift rates with the uses of existing values. We prove the detecting performance by implementing RSTS on MICA2 motes. Implementation result of RSTS shows 76.6% of detection rate against modified timestamps with adding 100 microseconds, and 100% of rate against over 1millisecond attacks. Thus, RSTS is a simple, efficient, and flexible solution of secure time synchronization against internally compromised nodes in wireless sensor network.

6.2 Future Works

In this thesis, we provide the secure time synchronization mechanism for single hop time synchronization. We would like to extend our secure mechanism to multi-hop synchronizations. And, we would like to analyze and enhance RSTS for reducing the misjudgment rate. Additionally, we would like to implement RSTS with the use of TinyOS version 2, which will support microsecond time unit, and we expect better performance in future implementation.

The estimated maximum and minimum relative clock drift rates are used on the secure time synchronization in this thesis. We hope to employ the estimated relative clock drift rates to carry out self-adjustment of sensors' clock for long live wireless sensor network.

REFERENCES

[1] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002. http://lecs.cs.ucla.edu/Publications.

[2] S. B. Moon, P. Skelly, and D. Towsley. Estimation and Removal of Clock Skew from Network Delay Measurements. Proc. IEEE INFOCOM, Vol 1, pp. 227–234,Mar. 1999.

[3] D. L. Mills. Internet Time Synchronization: the Network Time Protocol. IEEE Trans. Communications, Vol 39, no 10, pp. 1482–1493, Oct. 1991.and Computing (MobiHoc ’01), pp. 173–182, Oct. 2001.

[4] S. Ganeriwal, R. Kumar, andM. Srivastava. Timing-Sync Protocol for Sensor Networks. Proc. First Int. Conf. on Embedded Networked Sensor Systems, Los Angeles, California, Nov. 2003.

[5] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia, “Leap: efficient security mechanisms for large-scale distributed sensor networks,” in Proceedings of the 10th ACM conference on Computer and communications security. 2003, pp. 62–72, ACM Press.

[6] F. Cristian. Probabilistic Clock Synchronization. Distributed Computing, 3:146–158, Springer-Verlag, 1989.

[7] K. Arvind. Probabilistic Clock Synchronization in Distributed Systems. IEEE Transactions on Parallel and Distributed Systems, 5(5):474–487,May 1994.

[8] Sichitiu, M.L. and Veerarittiphan, C. Simple, accurate time synchronization for wireless sensor networks. IEEE Wireless Communications and Networking Conference (WCNC03), New Orleans, Louisiana, USA. March 16–20.

[9] D. M. Hawkins, Identification of Outliers, New York: Chapman and Hall, 1980.

[10] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. Computer Networks, 38(4):393–422, March 2002.

[11] Kusy, B. et al. Elapsed Time on Arrival: A simple, versatile, and scalable primitive for canonical time synchronization services, accepted for publication in Int. J. of Ad Hoc and Ubiquitous Computing, 2005

[12] Available at http://www.ieee-uffc.org/freqcontrol/quartz/vig/vigcateg.htm

[13] Song H., Zhu S., Cao G. Attack-Resilient Time Synchronization for Wireless Sensor Networks. MASS05, November 2005

[14] Ganeriwal S., Capkun S., Han C., Srivastava M. Secure Time Synchronization Service for Sensor Networks. Wise, September 2005

[15] M.D. Lemmon, J. Ganguly, and L. Xia. Model-based Clock Synchronization in Networks with Drifting Clocks. Proc. 2000 Pacific Rim International Symposium on Dependable Computing, Dec. 2000.

[16] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In ASPLOS X, San Jose, USA, October 2002.

[17] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. RSA CryptoBytes, 5(2):2--13, 2002.

[18] A Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar. SPINS: Security Protocols for Sensor Networks, Proc, of ACM Mobicom'01.

[19] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, and H.O. Marcy. Wireless Integrated Network Sensors: Low Power Systems on a Chip. In Proceedings of the European Solid State Circuits Conference, 1998.

[20] William Merrill, Lewis Girod, Jeremy Elson, Kathy Sohrabi, Fredric Newberg, and William Kaiser. Autonomous Position Location in Distributed, Embedded, Wireless Systems. In Proceedings of IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, CA, September 2002.

[21] Lewis Girod, Vladimir Bychkovskiy, Jeremy Elson, and Deborah Estrin. Locating tiny sensors in time and space: A case study. In International Conference on Computer Design ICCD, September 2002.

[22] Sundararaman, B., Buy, U., Kshemkalyani, D. Clock synchronization for wireless sensor networks: A Survey. Ad-hoc Networks, 3(3): 281-323, May 2005.

[23] Greumen. J. V., Rabaey, J. Lightweight time synchronization for sensor networks. In Proceedings of the Second ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), San Diego, CA, 2003.

[24] Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

[25] J. Hill and D. Culler. A Wireless Embedded Sensor Architecture for System-level Optimization. Technical report, U.C. Berkeley, 2001.

[26] Available at http://www.xbow.com/Support/Support_pdf_files/Getting_Started_Guide.pdf

[27] Available at http://www.tinyos.net/

[28] Available at https://mail.millennium.berkeley.edu/pipermail/tinyos-commits/2003-December/000790.html

[29] The nesC Language: A Holistic Approach to Networked Embedded Systems, David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler. In Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003.

[30] Available at https://mail.millennium.berkeley.edu/pipermail/tinyos-commits/2003-December/000791.html

[31] W. Zhuang. RLS algorithm with variable forgetting factor for decision feedback equalizer over time-variant fading channels. Wireless Personal Communications, Vol.8, No.1, pp. 15-29, August 1998.

BIOGRAPHICAL INFORMATION

Choi, Jae Sung is a graduate student in Computer Science and Engineering at the University of Texas at Arlington. He received his Bachelor in Computer Engineering from Kung Sung University, Busan, South Korea in 2003. His research interests include wireless sensor networks and mobile computing.