國立交通大學

網路工程研究所

碩 士 論 文

無線微型感測網路上之具防禦力的時間同步協定

Outlier-Filtered Time Synchronization Protocol for WSNs

研 究 生：簡浩洋

指導教授：謝續平　教授

中 華 民 國　九 十 六　年 六 月

無線微型感測網路上之具防禦力的時間同步協定

Outlier-Filtered Time Synchronization Protocol for WSNs

研 究 生：簡浩洋　　Student: Hao-Yang Jian
指導教授：謝續平 博士　　Advisor: Dr. Shiuhpyng Shieh

國 立 交 通 大 學
資 訊 工 程 學 系
碩 士 論 文

A Thesis
Submitted to
Institute of Network Engineering
College of Computer Science
National Chiao Tung University
In Partial Fulfillment of the Requirements
For the Degree of
Master
In

Computer Science

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

無線微型感測網路上之具防禦力的時間同步協定

研究生：簡浩洋　　指導教授：謝續平

國立交通大學　資訊工程學系

摘　要

對無線微型感測網路來說，時間同步是一個重要的前提，因為在眾多的應用程式中，皆需要建立在彼此的時間需一致之上，此類的應用有物體追蹤、速度測量、資料聚合和使用時間槽方式溝通的應用。有一些攻擊會降低現存協定的精確度。此類的攻擊有 pulse-delay attack、spoofing attack、replay attack 和 insider attack。目前已經有很多篇 PAPER 以安全的角度去設計時間同步協定，但他們並沒有辦法完整抵禦這些攻擊，或是只同步了時間差而沒有同步頻率差。此篇論文介紹了兩種有效率的傳遞方式，一個是使用群播的方式，一個是使用改進過的 μTESLA 的方式。最後我們提出了一種篩選演算法能濾掉 Outlier，我們使用了簡單的評分機制濾掉 Extreme outliers 然後使用線性回歸線來過濾 Mild outliers。在我們的實驗中，我們產生了一些 Outliers，然後使用我們的過濾演算法，我們觀察到當過濾比率比 Outlier 的比率大時，同步完的時間將不會受到 Outlier 的汙染且是非常準確的。

Outlier-Filtered Time Synchronization Protocol for WSNs

Student: Hao-Yang Jian Advisor: Shiuhpyng Shieh

Institute of Network Engineering
College of Computer Science
National Chiao Tung University

Abstract

Time synchronization is considered an important issue for wireless sensor networks (WSNs) where many applications, such as object tracking and data aggregation, rely on the synchronization of the clock of each node. Some types of attacks, such as insider attack and pulse-delay attack, can be used to decrease the precisions of time synchronization protocols. Recently, many time synchronization protocols were designed with security concern. Some of them cannot defend against these attacks, while others cannot synchronize the relative clock skews and offsets simultaneously or cannot support global time synchronization in multi-hop sensor networks. In this paper, we propose two efficient methods to reduce the overhead for propagating the global time. One uses authenticated multicast by inserting multiple MACs into a message. This method is more efficient than the other, but the amount of receivers must be less than eight. The other uses authenticated broadcast based on uTESLA and it doesn't limit the amount of receivers. The proposed methods can filter the outliers introduced by attacks and keeps the precision about the order of microsecond.

誌　謝

本篇論文的完成首先要感謝我的指導老師謝續平教授。感謝老師認真的指導以及給予許多寶貴的經驗與意見，也感謝老師不厭其煩的幫我修改文章，以致能有今天這份完整的畢業論文。其次要感謝我的父母，感謝他們多年來辛苦的栽培，沒有他們在背後支持，今日我將無法得以在此完成本篇論文，感謝他們總在我疲憊的時候給予我最多的鼓勵，在我徬徨的時候指點我正確的方向。此外要感謝分散式系統與網路安全實驗室的諸位學長姐學弟妹與同學們，感謝他們在我論文撰寫過程給予的諸多幫助，因為有他們的督促以及經驗交流，我才能順利的通過碩士論文的考驗。在論文寫作期間，我的老婆適逢懷孕時期，很感激她幫我分擔了大部分的責任，讓我能夠專心於研究之路，謝謝妳，老婆。

最後我要感謝的是所有在我求學過程曾經幫助過我的師長朋友，感謝所有人給予我的關懷和照顧，這篇論文是我至今完成過最棒的作品。

Table of Contents

Table of Figures

1. Introduction

Recent advances in technology have made low-cost, low-power wireless sensors a reality. Wireless Sensor Networking is an emerging research area with potential applications in environmental monitoring [11][12][13], surveillance, military, health [14][15] , and security. Such a network normally comprises a few sink nodes and a huge number of nodes, called sensor nodes. Each node is equipped with one or more sensors, an embedded processor, and a low-power radio. Typically, these nodes are linked by a wireless medium to perform distributed sensing tasks. This kind of sensor networks offers a monitoring capability in virtually any environment even if a wired connection is not possible or physical placement of the nodes is difficult. The sensor nodes are responsible for sensing the environment, such as temperature, humidity, sound, light, pressure, intruder detection, and location tracking. Sensor nodes will buffer the sensing data and then send it to sink node. A sink node is a more powerful sensor node and it is the control center of a sensor network where user could retrieve data gathered from sensor networks. Time synchronization is an important issue in WSNs. If each node's clock is different from each other, the sensing result would be useless because of an event including both sensing data and real timestamp. An attacker might launch different kinds of attacks to break the time synchronization protocol due to its importance.

1.1. Features of Wireless Sensor Network

The major differences among Wireless Sensor Networks and other wireless networks, such as Mobile Ad-hoc networks and cellular networks, are listed in

[16][17][18][19]：

1. Critical energy consumption：The small volume of sensor node causes the critical battery capacity. It seems to be impossible to recharge the batteries of sensor nodes. The energy consumption becomes the most critical issue of the design of sensor node.
2. Low communication bandwidth：Radio-frequency transmission is the only channel to transmit data, but it is the main power consumption on a node. The bandwidth of Wireless Sensor Networks is about 20 – 250kb/s and is relative low to the tradition wireless networks.
3. Limited computing power and memory space：Due to the small volume and low cost of each sensor node, the computing power and memory space are critically limited. There is only several kilo bytes to hundreds kilo bytes memory equipped on each sensor node, and the computing power ranges from 4MHz to 100MHz.
4. The large scales of deployment：Wireless Sensor Networks often consist of hundreds, even thousands of wireless sensor nodes. Those sensor nodes are deployed in a large wireless area for some monitoring task.
5. Highly damageable environment：Due to the low-cost design of wireless sensor devices, tamper resistance is beyond the concept. Each node is highly damageable under many external forces from the deploying environment.

The success of Wireless Sensor Networks is the appearance of tiny, lightweight network devices, called MICAz. MICAz is a low-power, tiny wireless measurement system which is designed specifically for deeply embedded sensor networks. Its wireless communication transceiver follows IEEE 802.15.4/ZigBee spec with 250

kbps data rate. The program memory is 128K bytes while data memory is 4K bytes. Power consumption of microprocessor is 8mA in active mode and less than 15 μ A in sleep mode while that of radio-frequency transceiver is 19.7mA in receive mode, 11mA in transmit mode, 20 μ A in idle mode and 1 μ A in sleep mode.

1.2. Issue of Time Synchronization

The reason why time synchronization in sensor networks requires more precisely than in traditional Internet applications is their close coupling with the physical world and their energy constraints. It's sometimes on the order of microsecond. For example, the target tracking applications using Kalman filter to estimate the target position, measuring the time-of-light of sound and distributing a beamforming array need the precise time. An event consists of timestamp and sensed data and sink node needs the timestamp to suppress redundant messages by recognizing duplicate events sensed by different sensors.

Network Time Protocol (NTP) is the current standard for time synchronization on the Internet. Though it has been deployed broadly and proven to be effective, secure and robust on the Internet, it is not suitable for Wireless Sensor Networks because of its energy consumption and lower accuracy about the order of millisecond. Global Position System (GPS) is the other approach to synchronize to external timescale and its accuracy could be about 200 nanoseconds relative to UTC. There are two reasons that a sensor is not suitable to equip GPS device. One is the energy consumption, because sensors usually do not have much power. The other is that GPS requires clear sky view, but many applications of sensors are inside of buildings, beneath dense foliage and underwater.

Many clock synchronization protocols for Wireless Sensor Networks have been

proposed over the years, but some of them are not designed with security in mind. These insecure protocols could be classified as three categories. The protocols in first category are precision driven; their idea is to maximizing the clock precision [5][6][7][9]. Second category is lightweight driven; their focus is on minimizing the power consumption [26][27][28]. Another category is scalability driven; the advantage of them is allowing user to trade off between convergent time and energy consumption [27][29][30]. Because without secure concern, all above protocols designed without security may suffer from many kinds of attacks like spoofing attack, replay attack, pulse-delay attack and insider attack.

1.3. Attacks for Time Synchronization Protocol

There are various attacks proposed for time synchronization for WSNs. Some of them could be defended against by using authenticated communication. Two of them, pulse-delay attack [20] and insider attack [21], cannot be solved using cryptographic techniques. They will affect the time of reception of message and mislead a receiver to adjust clock incorrectly.

1.3.1. Spoofing Attack

Spoofing attack means that an external attacker may fake (local) broadcast messages used for global synchronization or forge broadcast messages when relaying them to mislead the regular nodes. Sensor nodes usually use broadcast to communicate with each other. In hostile environments, broadcast must be authenticated to promise the authenticity and integrity of the broadcast messages. For defending spoofing attack, promising the integrity of the messages is essential, and currently there are two ways, digital signatures and TELSA-based approaches, could

provide broadcast authentication. However, both of approaches are not applicable for time synchronization in Wireless Sensor Networks. Such operations which perform public key cryptography on low-end sensor nodes still cost substantial computational and power resources and are subject to DoS attacks. The TESLA-based approaches use efficient symmetric cryptography to provide the broadcast authentication. However, the TESLS-based approaches require loose time synchronization between the sender and receivers, and thus cannot be used for global time synchronization directly when they are not synchronized. In our scheme, two approaches are proposed to promise the integrity of the messages. To perform these approaches, the assumption is that any nodes which want to communicate with the other node must share a unique pairwise key with it, so that the communications between them are authenticated. Such pairwise keys could be provided by key predistribution schemes proposed for sensor networks recently [3][4].

1.3.2. Replay Attack

Due to the property of wireless media, messages could be monitored by anyone. External attackers could buffer the messages and replay them after a period. If receivers accept this kind of messages, the result will be wrong. Fortunately, conventional solution for replay attack is padding a unique number like counter to messages. Essentially, the timestamp is a counter, so the duplicated messages could be found by checking if the timestamps in messages are the same or not.

1.3.3. Pulse-Delay Attack

Pulse-delay attack is like replay attack, but the attacker needs more effort to achieve this attack. Figure 1-1 shows the situation of pulse-delay attack. The sender

sends a message to the receiver at T_s and the receiver will receive the message at T_r. In normal situation, the relative equation is $T_r = T_s + \delta + d$. δ is the clock offset between the sender and the receiver and d is the end-to-end delay between the sender and the receiver.

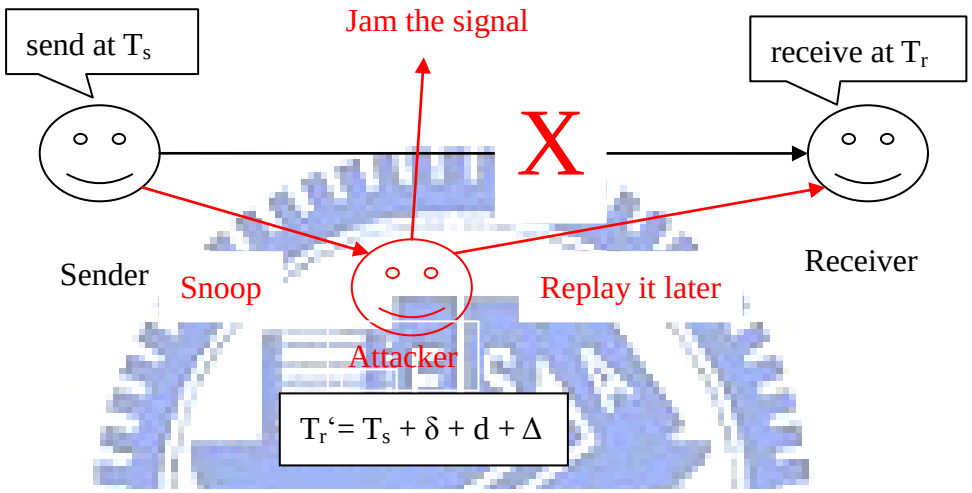


Figure 1-1: Pulse-Delay Attack

When the sender sends a message, the attacker not only snoops, but also lets the receiver miss the message by jamming the channel. After a period, the attacker replays it to receiver. Because the attacker does not change the content of message, the receiver cannot find any exception by cryptographic techniques. If pulse-delay attack occurs, the relative equation will change to $T_r' = T_s + \delta + d + \Delta$. Δ is the delay performed by the attacker. In two-way communication, the sender could detect this attack by Secure Pairwise Synchronization (SPS) [20]. But in one-way transmission, because the receiver cannot know the occurrence of this attack, they cannot predict and fix the Δ.

1.3.4. Insider Attack

The definition of insider attack is the attack performed by compromised node. Here, the compromised nodes possess valid cryptographic keys. Most of the time synchronization protocols assume benign environments; however, sensor nodes may be compromised by malicious intruders to certainly attack the clock synchronization protocols due to the importance of synchronized clock time. Though authentication could be used to defend against external attacks, clock synchronization may still suffer from compromised nodes. The compromised node may report the incorrect clock information. In pairwise clock synchronization, the compromised node has limited impact on single-hop neighbor nodes. However the global time synchronization using multihop paths is vulnerable to compromised nodes because a compromised node in the path could introduce arbitrary errors and affect the nodes which are multi-hops away from it.

We assume the compromised nodes may collude together to disrupt clock synchronization. They may cheat normal nodes by sending incorrect clock information. If they don't cooperate, the attacks could not cause maximum damage and even weaken them. Our goal is that even if a certain number of compromised nodes collude together to disrupt clock synchronization, each normal node could still synchronize its local clock to the global clock.

1.4. The Weakness of Flooding Time Synchronization Protocol

Flooding Time Synchronization Protocol (FTSP) [9] is the newest one with many advantages comparing to others. It decomposes the transmission delay exactly and adjusts sending timestamp in advance. The techniques FTSP used are the mac layer time-stamping which eliminates the most uncertain delay introduced by accessing the

communication channel and linear regression to predict the relation between local clock and global clock, thus, FTSP could be more precise than others. FTSP uses a root node to broadcast its local clock considered as global clock to all other nodes, and other nodes receive the syncMsg from the root and synchronize it local clock to global clock. While receiving a syncMsg, the receiver calculates the offset between local clock and the estimated global clock of sender. Then the receiver calculates its clock skew by using linear regression on a set of these offsets versus the time of reception of the messages. After a node was synchronized, it sent the estimated global clock to other nodes that could not communicate with root directly. On account of the resource-constrained hardware and the relation between global clock and local clock being linear in a short term, every node keeps 8 reference points only.

There are two problems in FTSP. When the sender propagates the syncMsg, the receivers cannot promise the integrity of it. This introduces the external attackers could launch spoofing attack to break the time synchronization protocol. The other problem is that the receiver does not filter any outlier. In statistics, an outlier is an observation that is numerically distant from the rest of the data. Statistics derived from data sets that include outliers will often be misleading. The outlier could be generated due to above mentioned attacks or systematic error.

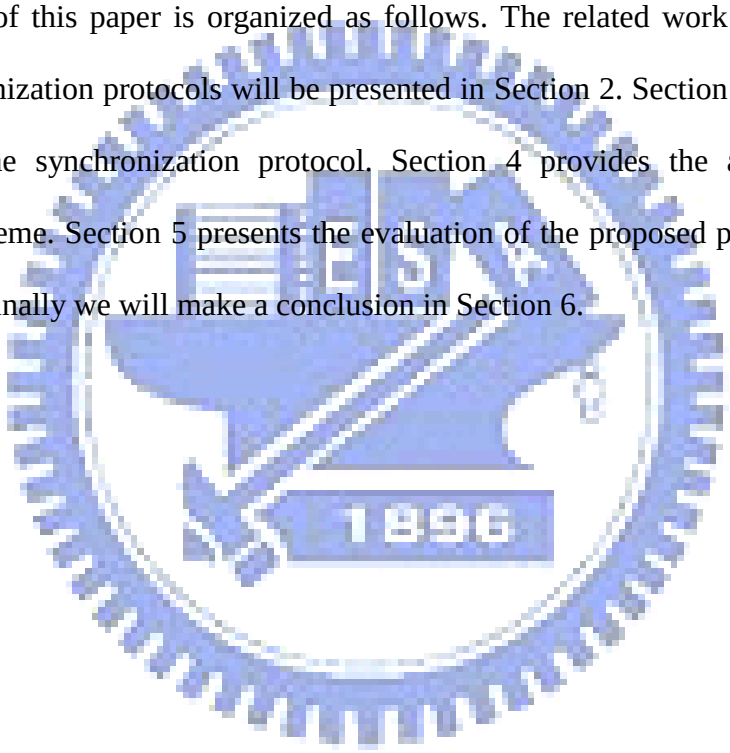
1.5. Contribution

We propose a resilient time synchronization protocol based on FTSP [9]. Our approach works normally even when attacks occur. The attacks which could be defended against are pulse-delay attack, insider attack, spoofing attack and replay attack. We cannot handle Denial-of-Service attacks that completely jam the communication channel. All the same, no existing protocol could defend against such

extreme DoS attacks. Our scheme proposes two methods to reduce the overhead of communication and promises the integrity of messages simultaneously. A neat method is employed to filter outliers in the collected datasets because of the resource constraints of sensors. After filter and calculation, the nodes would get the relation between local clocks to global clock.

1.6.Synopsis

The rest of this paper is organized as follows. The related work of the existing time synchronization protocols will be presented in Section 2. Section 3 describes our proposed time synchronization protocol. Section 4 provides the analysis of the proposed scheme. Section 5 presents the evaluation of the proposed protocol through simulation. Finally we will make a conclusion in Section 6.



2. Related Work

Time synchronization is important in many systems, both wired and wireless, and a large number of time synchronization schemes exist. Well-known synchronization schemes include GPS [2] and the Network Time Protocol (NTP) [1]. There are many insecure and secure time synchronization protocols for WSNs, but they are still insufficient in some aspects.

2.1. Conventional Schemes

Wireless Sensor Networks pose a number of challenges beyond traditional network systems. Elson and Romer [32] describe the differences quite exhaustively and the other detailed design principles are described by Santashil et al.[30] Network Time Protocol (NTP) is the current standard for synchronizing clock on Internet, but it is not suitable for Wireless Sensor Networks because of some constraints. One is that WSN applications need higher precision such as the order of microsecond because of their close coupling with the physical world, but the precision of NTP is about the order of milliseconds. The other reason is that sensor node is usually resource-constrained and NTP is not energy efficient although it has been widely deployed and proved to be effective and robust on the Internet. Global Position System (GPS) devices could synchronize the time to external timescale and its accuracy could be about 200 nanoseconds relative to UTC. However GPS is not available in places such as inside building and underwater and it might also be power-consumed and expensive relative to resource constraint sensor nodes. Thus, both protocols are not suitable for Wireless Sensor Networks.

2.2. Insecure Time Synchronization Protocols for WSNs

These insecure protocols could be categorized as three parts. The protocols in first category are precision-driven and focus on maximizing the clock precision [5][6][7][9]. The reference broadcast synchronization (RBS) scheme [5] is the first work addressing the time synchronization issue in sensor networks; however it could only synchronize multiple receivers in a local region. Later, another scheme, called timing-sync protocol for sensor networks (TPSN) [7], was proposed and it could achieve network-wide time synchronization. The most precise scheme, called Flooding Time Synchronization Protocol (FTSP) [9], is the one with many advantages comparing to others. It decomposes the transmission delay exactly and adjusts sending timestamp in advance. The techniques FTSP used are the mac layer time-stamping which eliminates the most uncertain delay and linear regression to predict the relation between local clock and global clock, thus, FTSP could be more precise than others. Another advantage of FTSP is that it supports dynamic network topology and this makes FTSP more complete. Unfortunately, FTSP is an insecure scheme because it was designed without security concern. Two types of attacks have been proposed, namely pulse-delay attack and insider attack introduced by compromised nodes. Both of these attacks form the outliers of collected syncMsg, but FTSP cannot filter them. In FTSP, when attacks occur, some of the reference points which contain a pair of global and local timestamps referring to the same time instant become outliers and introduce serious error. Another problem is that FTSP propagate timestamp by broadcast, but it does not support message authentication; so external attackers could forge the packets with an arbitrary syncMsg. Our goal is to filter the outliers which are further away from their expected values than what are deemed reasonable.

Second category is lightweight driven; their concentration is not to maximize accuracy, but to minimize the complexity to achieve a given precision [26][27][28]. Thus, the needed synchronization accuracy is assumed to be given as a constraint, and the target is to devise a synchronization algorithm with minimal complexity to achieve a given precision. This approach is supported by the claim of the authors that the maximum time accuracy needed in sensor networks is relatively low (within fractions of a second), so it is sufficient to use a relaxed, or lightweight, synchronization scheme in sensor networks.

Another category is scalability driven [27][29][30]. These protocols consider clock synchronization might not be necessary at all times, except during sensor reading integration. Providing clock synchronization all the time will be a waste on the limited resources of sensors. For saving resources, the nodes re-synchronize only when there is a need for synchronization.

2.3.Secure Time Synchronization Protocols for WSNs

There are some studies for secure time synchronization in sensor network proposed recently [20][21][22][23][31], but there are some insufficiencies among them. The insufficiency of the Secure Pairwise Synchronization (SPS) proposed by Ganeriwal et al. [20] is that it just aborts the action when detecting the attacks and it cannot achieve the goal of time synchronization. Manzo et al. discussed the attacks against time synchronization protocols and proposed some countermeasures [21]. But it still suffers from pulse-delay attack and it doesn't resolve the conflict when using the μ TESLA-based broadcast authentication which requires loose time synchronization. Sun et al. proposed a resilient time synchronization protocol whose focus is on the defense of compromised nodes [23], but it also suffers from pulse-delay attack. The

problem of Song et al. proposed two methods for detecting and tolerating delay attacks [22], but the insufficiency is that it doesn't support global time synchronization in multi-hop sensor networks. The latest scheme, called TinySeRSync [31], has solved all existing attacks. The concept of TinySeRSync for solving insider attack is to choose the median from 2t+1 data. It adopts the Secure Pairwise Synchronization (SPS) [20] with a slight modification to deal with pulse-delay attack and wormhole attacks. To avoid substantial communication overhead as well as frequent message collisions in dense sensor networks, it designs a local authenticated broadcast for the propagation of global synchronization messages, effectively harnessing the broadcast nature of wireless communication. The insufficiency of TinySeRSync is that the sensor cannot get the clock skew between its local clock and the global clock to compensate the constant clock drifts. Due to the effect of clock skew, the nodes need to resynchronize frequently to maintain certain precision. The more messages a sensor transmits, the more power it consumes. For saving power, maximizing the interval of resynchronization period by compensating the constant clock drifts is necessity. Finally, the summaries of these insufficiencies of secure protocols are in Table 1.

Table 1: The insufficiencies of existing secure time synchronization protocols

Paper	**Existing problems**
Secure time synchronization service for sensor networks [20]	Only abort the action when detecting the attacks
Time synchronization attacks in sensor networks [21]	Cannot defend against pulse-delay attack
Attack-resilient time synchronization for wireless sensor networks [22]	Can't support global time synchronization in multi-hop sensor networks
Secure and resilient clock synchronization in wireless	Cannot defend against

sensor networks [23]	pulse-delay attack
TinySeRSync: Secure and resilient time synchronization in wireless sensor networks [31]	Only synchronize the initial offset instead of clock skew

3. Proposed Scheme

In this section, we present our Resilient Time Synchronization Protocol, namely RTSP. Some notations and definitions are mentioned at first. The offset of two clocks is the time difference between them, while the clock skew is the frequency difference between them. Global time is considered as the time of sink node. A dataset consists of two timestamps; one is the global time estimated by sender and the other is the time of reception of packet. The outlier is the dataset which is further away from its expected value than what is deemed reasonable. It could be introduced by critical error or attacks. The neighbors of a node are those nodes which are one hop away from it. The summary of these notations and definitions is in Table 2. The assumption is mentioned following. In Section 3.2, the overview of RTSP is described. The remainders of Section 3 would explain the details of two phases in RTSP.

Table 2: Notations and Definitions

Clock Offset	The clock offset is the time difference between two clocks. It consists of initial offset and clock skew.
Clock Skew	The clock skew is the frequency difference between two clocks.
Global Time	It is considered as the time of sink node.
Dataset	It consists of two timestamps, one is the global time estimated by sender and the other is the time of reception of packet.
Outlier [33]	The dataset which is further away from its expected value than what is deemed reasonable.
Neighbor	The neighbor is one hop away from A.

F	Hash function
$F^j(K)$	hash K j times

3.1. Assumption

A sensor network consists of a large number of resource-constrained motes such as MICA series of motes. There is a trusted source node that is well synchronized to the external clock, for example, through a GPS receiver. The goal is to synchronize the clocks of all the sensor nodes in the network to that of the source node. The assumption of the single, trusted source node is to simplify the discussion in this paper.

Before a node start to synchronize its local time to global time, it must share a secret pairwise key with its neighbors by some approaches which could defend against insider attacks [3] and doesn't require the loose time synchronization. The secret pairwise key is utilized to create the Message Integrity Code (MIC) added into the sending packets to defend against the Sybil attacks [24][25], that one node presents multiple identities to defeat typical fault tolerant mechanisms.

3.2. Basic Scheme

In our approach, a sink node needs to propagate its local time to each sensor. The global time of the network is considered as the sink node's time. In previous approach FTSP [9], it needs a root election procedure, but the sink node is essential in a WSNs. Because the sink node is the data fusion of entire sensor network, the network cannot work on the whole without it. Base on this hypothesis, we decide to employ the sink node as our root in time synchronization protocol permanently. At the start of each synchronization procedure, the root sends its clock information to its neighbors. It is

some different with FTSP approach. FTSP uses the broadcast to communicate with its neighbors. In our case, the nodes propagate global time through three different methods because the integrity of messages cannot be promised in FTSP. After a node was synchronized, it continued to send the estimated global time to its neighbors. The receiver records the corresponding local time at the reception of the packet. An estimated global time by sender and a timestamp from its local clock constitute a dataset. The receiver filters some outliers and uses the remainder datasets to calculate the relative clock skew and offset when receiving enough datasets. After working out the clock skew of its local clock and initial offset between local clock and global clock, it was synchronized because it could map its local time to global time. Then it could propagate its estimated global time to next level neighbors. In the following sections, we will present the detail of propagating global time and filter of outliers.

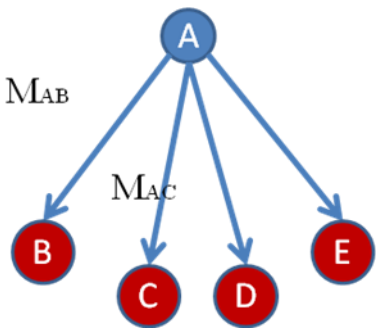
3.3. Phase I: Propagate Global Time to One-hop Neighbors and Promise the Integrity

In this phase, a synchronized node needs to send its estimated global time to its one-hop neighbors. The receivers only verify the integrities of messages, but not the timeliness. The problem is handled in phase II. The sink node sends some packets including its clock information to its one-hop neighbors to help them to adjust their clocks initially. Because there is only one source at start of each round, the sink node must send at least eight packets to let its neighbors collect adequate datasets to work out their clock skews and offsets. The minimal count of datasets is eight because FTSP [9] uses an eight entries regression table to store the datasets and is good enough to meet the precision about the order of microsecond. But there might be some outliers in the datasets, we enlarge data table from 8 entries to more than 16

entries to tolerate them. After the neighbors of the sink node synchronize to the clock of sink node, the nodes which are two hops away from the sink node could receive estimated global time from them. If nodes receive the messages including the same global time, nodes simply omit duplicate message to prevent replay attack. There are huge methods about sending estimated global time to neighbors. We will propose three of them.

3.3.1.Authenticated Communication with Unicast

The first method is the simplest method. Each node uses authenticated unicast communication to propagate synchronization messages. The sending messages include the estimated global time and the message authentication code (MAC). The MAC is generated using the data and the secret pairwise key only shared by sender and receiver, so it could protect both message's integrity as well as its authenticity, by allowing receiver to detect any changes to the message content. External attackers cannot generate illegal messages or tamper messages because they do not have the pairwise key with normal nodes. The problem of this method is that it may introduce substantial communication overhead as well as frequent message collisions in dense sensor networks.



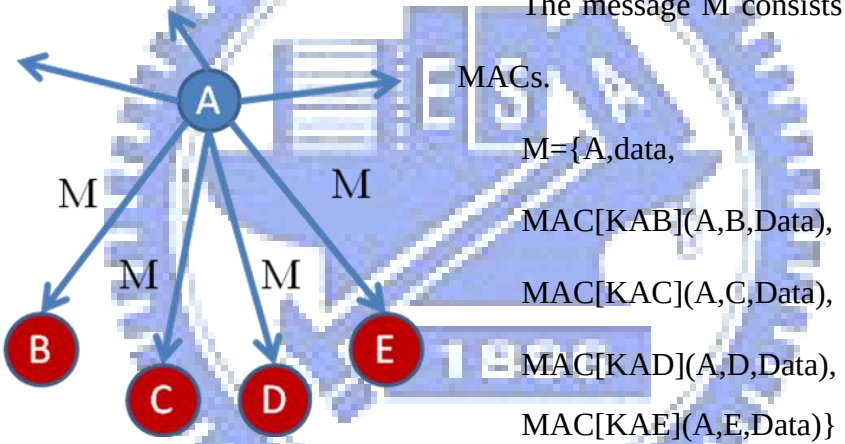
$M_{AB}=\{A,B,Data\ ,MAC_{KAB}(A,B,Data)\}$

$M_{AC}=\{A,C,Data\ ,MAC_{KAC}(A,C,Data)\}$

Figure 3-1: Communication with unicast

3.3.2.Authenticated Communication with Multicast

The second method is to use broadcast to propagate the estimated global time in Figure 3-2. If one node wants to send message to its n neighbors, the messages will consist of the data and n MACs. Each MAC is generated by data and each pairwise key shared by sender and neighbors, separately. The receiver could authenticate the message by comparing each MAC with the MAC computed with the secret pairwise key with the sender. The limit of this method is that n must be less than eight because there is not enough time to insert the MAC before the transmission of the MAC due to the delay introduced by the MIC calculation [19][23].



The message M consists of data and n MACs.

M={A,data,
MAC[KAB](A,B,Data),
MAC[KAC](A,C,Data),
MAC[KAD](A,D,Data),
MAC[KAE](A,E,Data)}

Figure 3-2: Broadcast the message with n-MACs

3.3.3.Authenticated Communication with Broadcast

The last method also uses broadcast to propagate the estimated global time. It works something like *u*TESLA, but it resolves the conflict between the goal of achieving time synchronization and the fact that *u*TESLA requires loose time synchronization. Before the sender sends packets, it generates a random key K_i and uses K_i to construct a key chain by repeatedly hashing it. For example, K_{i-1} is the hash of K_i and K_0 is called the commitment of the key chain. Before transmitting packets, a

sender must send the key commitment to the receivers using authenticated unicast communication. So the integrity of key commitment is assured. Then the sender continues to broadcast next packet including MAC generated by the data and next key. To authenticate this packet, the receiver must buffer it and then authenticate the included key, namely K_j, while receiving more packets. When the receiver receives K_{j+n} at latter packets, it hashes K_{j+n} for n times to get the K_j' and checks if K_j is legal or not by comparing K_j' to K_j. If K_j is legal, the K_j could be used to verity the integrity of that packet.

Table 3: Local broadcast

Sender A	$\{K_0$,MAC_{KAB} ($K_{0)}\}$,Unicast	{Data ,K_1, MAC_{K1} $(Data_{)}\}$ Broadcast	{Data ,K2, MAC_{K2}(Data$_{)}\}$ Broadcast	{Data ,K3, MAC $_{K3}$ $(Data_{)}\}$ Broadcast	{Data ,K4, MAC $_{K4}$ $(Data_{)}\}$ Broadcast
Receiver B	$\{K_0$,MAC_{KAB} ($K_{0)}\}$,Unicast	{Data ,K_1, MAC_{K1} $(Data_{)}\}$ Broadcast if ($F(K_1)=K_0$) accept	Lost	{Data ,K_3, MAC_{K3} $(Data_{)}\}$ Broadcast If($F^2(K_3)=K_1$) accept	{Data ,K_4, MAC_{K4} $(Data_{)}\}$ Broadcast If($F(K_4)=K_3$) accept

3.4. Calculate the Relation between Local Clock and Global Clock

When a node collects ample datasets, it could utilize the least square to calculate the clock skew and initial offset between its local clock and global clock. The relationship between global time and local time could be described as Equation 1.

$C_r(t)=b_{rs}C_s(t)+a_{rs}$ Equation 1

b_{rs} is the relative clock skew and a_{rs} is the initial offset between global clock and local clock. If the realistic b_{rs} could be computed, the node could periodically compensate the drift caused by b_{rs} to keep the clocks more precise. Thus the resynchronization doesn't need to execute too frequently. The goal of time synchronization is not only making nodes have the same clock value in one instant by eliminating a_{rs} , but also periodically correcting the drift caused by b_{rs} a to keep the clocks synchronized.

For convenience, let us convert Equation 1 to Equation 2.

$Y = \alpha + \beta X + U$ Equation 2

α is equal to a_{rs}, β is equal to b_{rs} and U is the error due to some reasons introduced above. Then we want to find a linear equation to minimize ***U*.** The equation could be derived to Equation 3. To minimize Q, we make the partial differential of Equation 3 to Equation 4 and Equation 5.

$$Q(\alpha,\beta)=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\alpha-\beta X_i)^2 \quad =\frac{1}{n}\sum_{i=1}^{n}U_i^2. \qquad \text{Equation 3}$$

$$\frac{\partial}{\partial\alpha}Q(\alpha,\beta)=-2\frac{1}{n}\sum_{i=1}^{n}(Y_i-\alpha-\beta X_i)=0 \qquad \text{Equation 4}$$

$$\frac{\partial}{\partial\beta}Q(\alpha,\beta)=-2\frac{1}{n}\sum_{i=1}^{n}(Y_i-\alpha-\beta X_i)X_i=0 \qquad \text{Equation 5}$$

From Equation 4 and Equation 5, α and β are obtained**.** That means when a node collects adequate datasets, it could make use of x and y series to compute α and β, representing the initial offset and relative clock skew of local clock.

$$\hat{\beta}_n=\frac{\sum_{i=1}^{n}(X_i-\bar{X}_n)(Y_i-\bar{Y}_n)}{\sum_{i=1}^{n}(X_i-\bar{X}_n)^2},$$

$$\hat{\alpha}_n=\bar{Y}_n-\hat{\beta}_n\bar{X}_n.$$

3.5. Phase II: Filter Outliers and Calculate Local Clock's Skew and Offset

Before we detail this phase, let us see the different situations in benign environment and hostile environment. After phase I, a node could collect the normal datasets, such as that plotted as Figure 3-3. Because of the uncertain delay, real local clock offset cannot be got directly. The calculated offset is the original offset and delay. We just subtract that from the delay to get the real offset. The communication delay does not influence the local clock skew, so we do not need to reprocess the calculated value.

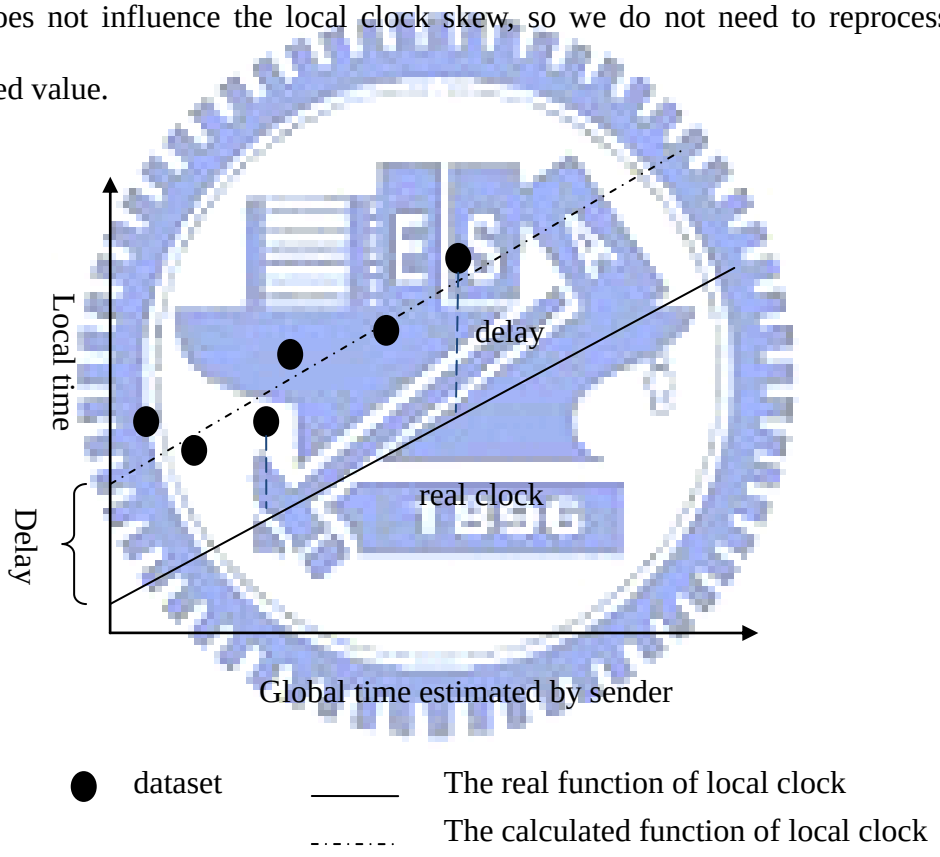


Figure 3-3: Datasets in normal situation

But at a hostile environment, the distribution of the collected datasets may become that in Figure 3-4 and in Figure 3-5, separately. The attacker could launch some attacks for time synchronization to pollute the result and mislead the node to synchronize wrong clock. Although Figure 3-4 and Figure 3-5 represent the outliers introduced by different attacks, the node couldn't differentiate between insider attacks

and pulse-delay attacks.

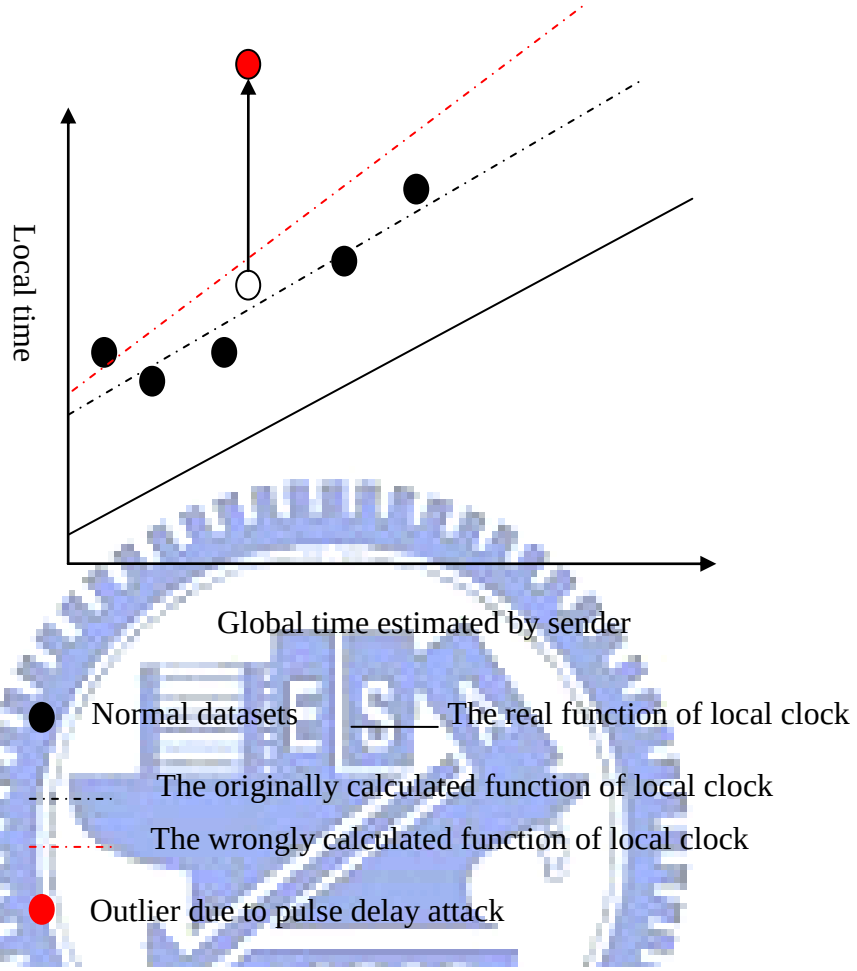


Figure 3-4: The situation of pulse delay attack

When pulse-delay attack occurs, the packet received is delayed. In Figure 3-4, the dataset moves toper than its expected value because the attacker postponed our reception of that packet. From Figure 3-4, we notice these outliers could perform serious error in the result.

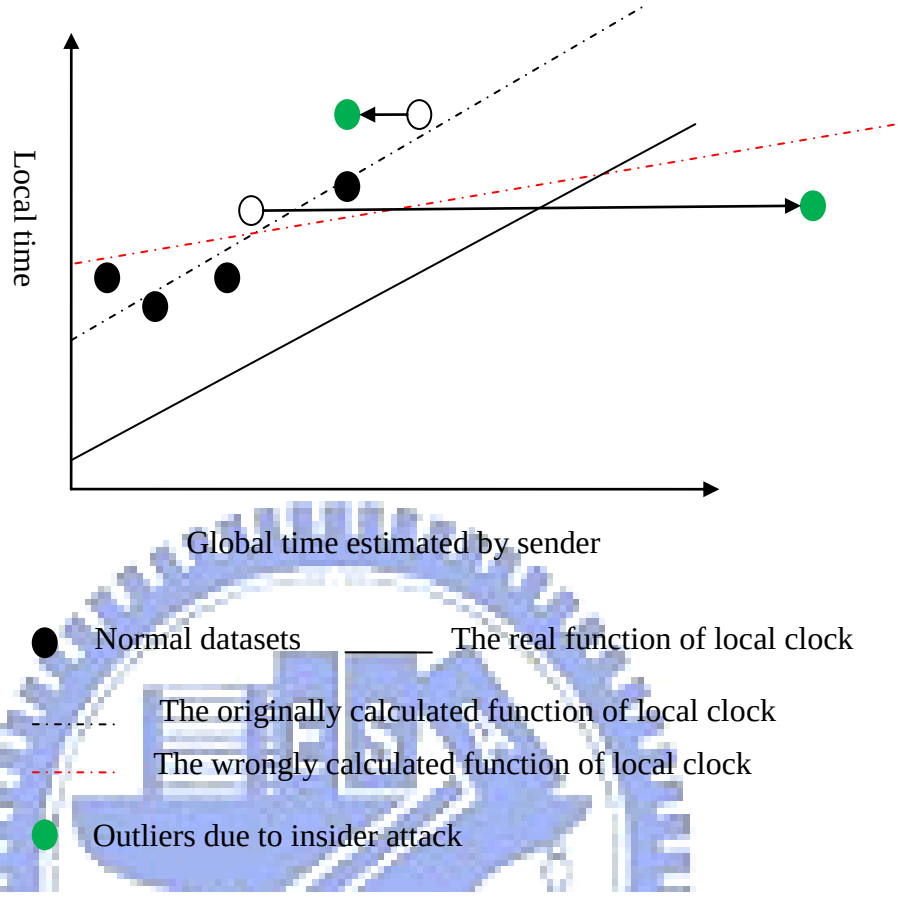


Figure 3-5: The situation of insider attack

When another attack, called insider attack, occurs, the global time we received is incorrect. In Figure 3-5, the outliers move righter or lefter than expected value because the compromised node would report bigger or smaller global time. If the outliers tend towards any side on average, it cannot warp our calculated result, so we assume the outlier will tend towards a specific side. Traditional noise filter like "Kalman filter" is not suitable for our situation, because it assumes that the observed noise is zero mean Gaussian white noise with covariance R_k. But the outliers introduced by attacks are not. V_k is the observed noise and represented as Equation 6.

$$V_k \sim N(0, R_k) \qquad \text{Equation 6}$$

We understand the truth from Figure 3-3, Figure 3-4 and Figure 3-5 that if we use all the datasets to calculate the relative clock skew and initial offset, we may get

the wrong answer due to the existence of outliers. The importance is that both pulse-delay attack and insider attack perform the same result, so we use the same filter method to filter outliers and use the remainder datasets as the input of Least Square Method.

3.5.1.Score Filter

The first step to filter the outliers is to calculate the score of each dataset. The purpose of score is to find the relative equation posed by most datasets and filter the outliers further away from this relative equation. The higher score means this dataset is more likely normal. The idea is that the relative equation consists of most datasets and we believe that the amount of normal datasets is more than that of outliers. There are two scalable arguments are n, the number of collected datasets, and m, the filter ratio (e.g., 0.5). Assuming we collect n datasets, for dataset d_1 (x_1 , y_1), x_1 is the estimated global time by sender and y_1 is the timestamp of reception of message, if d_2 satisfies the relation with d_1, the score of d_1 increases one. Figure 3-7 presents the normal bound of a dataset. δ is the variance of the predictive error. The normal relations are $yj - \delta - yi \,/\, xj - xi \le MaximumSkew$ and $yj + \delta - yi \,/\, xj - xi \ge MinimumSkew$.

Because the clock skew differences of the crystals used in Micaz motes introduce drifts up to 40μs per second, we consider 1.00004 and 0.99996 as the maximum and minimum skew. The upper line is a line through d_1 whose slope is 1.00004 and the upper bound is shift up this line with the variance of predictive error; the lower line is a line through d_1 whose slope is 0.99996 and the lower bound is shift down this line with the variance of predictive error. The pseudo-code describing how to generate the bound of dataset is presented at line 8-17 in Figure 3-7. After

calculating each dataset's score, we will insert the datasets to a collection S from maximum score to minimum score until the count of collection S is more than $\lceil n * (1 - m) \rceil$. Then the collection S continues to be the input of regressive filter.

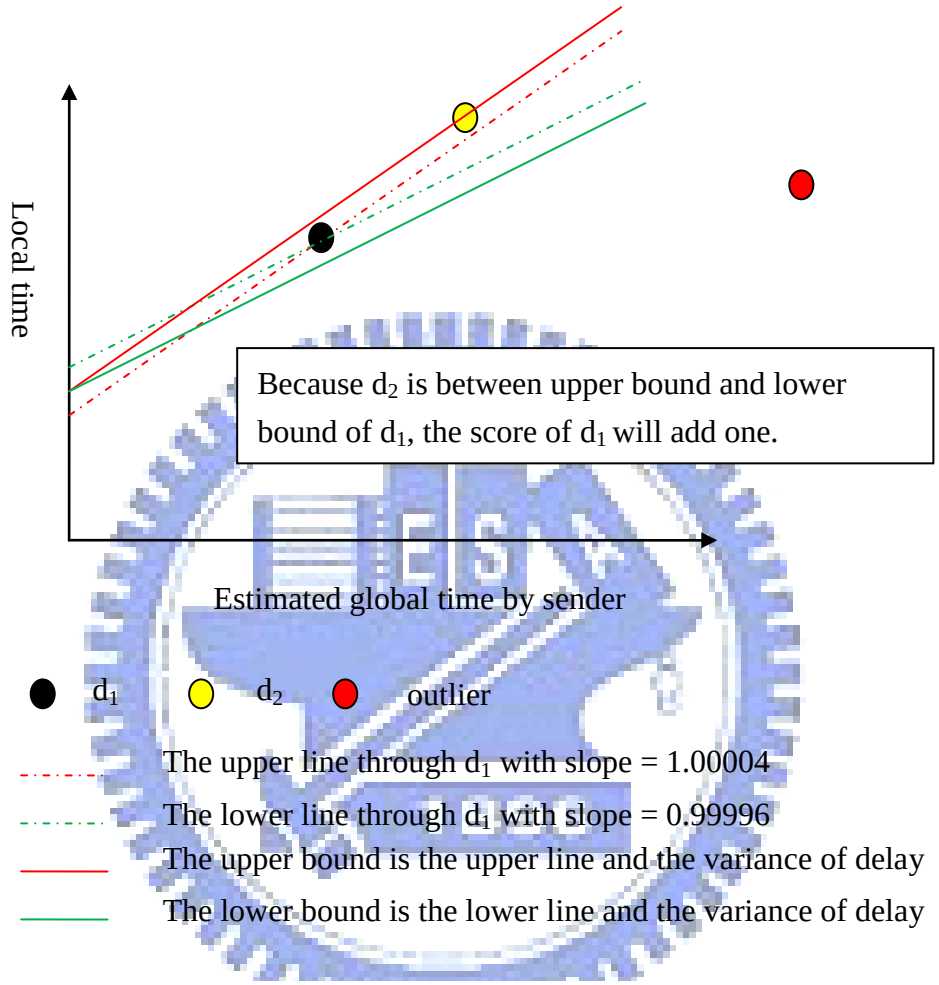


Figure 3-6: Calculating the Score

```
ScoreFilter(x,y)
1    for i ← 1 to count of x
2        do   for j ← 1 to count of x
3             △X ← Xj - Xi
4             if △X > 0
5                  then UpperY ← yi + △X * MaximumSkew + DelayVariance
6                       LowerY ← yi + △X * MinimumSkew - DelayVariance
7                  else UpperY ← yi + △X * MinimumSkew + DelayVariance
8                       LowerY ← yi + △X * MaximumSkew - DelayVariance
9
```

```
10              if    y_j ≥ LowerY and y_j ≤ UpperY
11                    then increase Score_i
12
13
14          SortScore ← SortByDesend(Score_i)
15          for NowScore ← SortScore(1) to SortScore(n)
16                do    for each Score ϵ SortScore
17                      do    if Score_i = NowScore
18                                  then Add d_i to PickUpDatasets
19                      if count of PickUpDatasets > n * m
20                            do return
```

Figure 3-7: Scoring each dataset and filter some outliers.

3.5.2. Regressive Filter

After score filter, maybe there are still some outliers included in that collection S because their scores are equal to some normal datasets. In this function, we will iterate to compute the regression line and delete the dataset which is farthest away from the regression line until the count of collection S equals to $\lceil n * (1 - m) \rceil$. The pseudo-code of regressive filter is presented in Figure 3-8. After regressive filter, we could consider the remainder datasets as normal datasets and use them to calculate the skew and offset of local clock.

```
RegressiveFilter(PickUpDatasets)
1     While Count of PickUpDatasets > n*m
2           do    Line=LeastSquare(PickUpDatasets);
3                 for i ← 1 to Count of PickUpDatasets
4                       do    NowDistance ← GetDistance(x_i,y_i,Line)
5                             if    NowDistance > MaxDistance
6                                   then MaxDistance ← NowDistance
7                                   MaxIndex ← i
8                 PickUpDatasets.Remove(MaxIndex)
```

Figure 3-8: Filter the dataset which is farthest away from the regression line.

4. Analysis

In this section, we will analyze the transmission overhead of three different methods to propagate global time in phase I. Figure 4-1 represents the efficiencies of different methods. Then we will discuss the security analysis of phase II.

4.1. Performance Analysis

We assume the count of one node's neighbors is n and n must be more than four. This could let one node collect more than 16 datasets. Each node would send four packets to its neighbors. A packet consists of Preamble (4 bytes), SFD (1 byte), Length (1 byte), FCF (2 bytes), DSN (1 byte), Address (1~20 bytes), Payload (n bytes) and FCS (2 bytes). The payload is the timestamp and its length is 8 bytes. So the size of a packet is 27 bytes.

First method using authenticated unicast needs 4 * n packets. A node must transmit 4 * n * 27 bytes. Second method using broadcasting multiple MACs needs 4 packets. The size of payload is bigger, because it consists of multiple MACs. The length of MAC is 2 bytes. A node must transmit 4 * (27 + n * 2) bytes. Third method using broadcast needs to transmit n+4 packets. The first n packets including key commitment are transmitted to its neighbors using authenticated unicast and next 4 packets are broadcasted to propagate global time. A node must transmit (n+4) * 27 bytes. We summarize the overhead of communication in Table 4.

Table 4: The overhead of transmission in three different methods (n is the count of neighbors)

Propagating methods in phase I	Transmitted bytes (n ≥ 4)
Unicast method	108 * n
Broadcast with multiple MACs	108 + 8 * n
*u*TESLA-based Broadcast	108 + 27 * n

The relation of neighbor's count and the needed bytes was plotted in Figure 4-1. From this figure, we could understand when the amount of receivers is less than eight, the performance of broadcasting multiple MACs is best. If neighbor's count is more than eight, the uTESLA-based broadcast is more suitable.

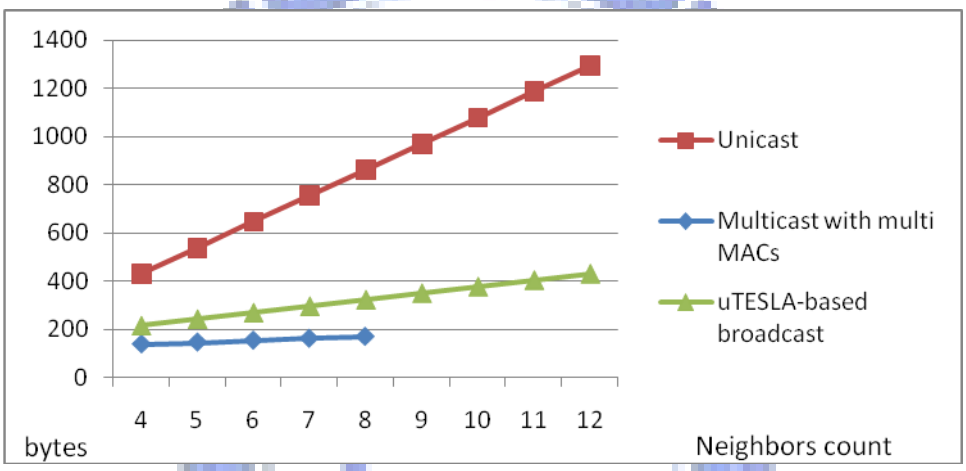


Figure 4-1: The relation between transmitting bytes and the count of neighbors

4.2.Security Analysis

The goal of phase I is to promise the integrities of packets and the authentication of communication. We will prove that the attackers cannot modify the packets or spoof the legal nodes. In the phase II, what we want to prove is when the ratio of outliers is smaller than half, our approach works successfully. The definitions are mentioned at first and we will prove that using the relation between the scores and amount of datasets.

Phase I: A node receives n datasets from C_m senders which are one-hop away from it. Nodes are defined as N_i, where i = 1… n. The key K_{ij} is only shared by N_i and N_j. The key pool owned by N_i is defined as KP_i. A *Message Authentication Code* is a family of functions *f1* of $\{0,1\}^k \times Dom(f1)$ to $\{0,1\}^l$, where *Dom*(*f1*) denotes the domain of *f1*. In this paper, $Dom(f1) = \{0,1\}^{\leq L}$. For $K \in \{0,1\}^k$ and $M \in \{0, 1\}^{\leq L}$, let $\sigma = f1(K,M)$. We refer to σ as the tag or MAC of *M*. The message forged by attackers is denoted as M_f. M_{ij} means the message which N_i sends to N_j. Theorem 1 states that different secret key cannot generate the same hash value.

Theorem 1: $\forall K, \exists K_{ij} \wedge K_{ij} \in KP_i \wedge K_{ij} \in KP_j, \sigma = f1(K_{ij},M_{ij}) \neq \sigma' = f1(K , M_{ij}), K \notin KP_k$

Proof.

$\because K_{ij} \in KP_i \wedge K_{ij} \in KP_j, K_{ij} \notin KP_k$

$\rightarrow \sigma = f1(K_{ij},M_{ij}), \sigma \neq \sigma' = f1(K , M_{ij}), K \notin KP_k$

$\rightarrow N_k$ cannot generate $\sigma = f1(K_{ij},M_{ij})$

$\therefore N_k$ can't forge or spoof the M_{ij}. □

Because the secret key is only shared by two nodes, from theorem 1, other nodes cannot generate the same hash value to spoof or forge the packets.

Phase II: A node receives n datasets which are defined as D_i, where i = 1…n. The amount of outliers denoted as DO is C_o and the amount of normal datasets denoted as DN is C_N. All datasets is denoted as DA. Outlier ratio m means the C_o divides n and the filter ratio r means how many datasets we will filter. We define the score of each

dataset as $S(D_i)$. We denote $F(D_i)$ as the instant of filtering the D_i and the $Dis(D_i)$ means the distance between D_i and the regression line. Theorem 2 states that when the amount of normal datasets is more than that of outliers, the scores of normal nodes are bigger than those of outliers.

Theorem 2: $DN \wedge DO = \varphi$, $D_i \in DO \wedge D_i \in DA$, $D_j \in DN \wedge D_i \in DA$, if $C_N > C_o$ $\rightarrow F(D_i) \geqq F(D_j)$

Proof.

$\because C_N > C_o$

$\rightarrow S(D_i) > S(D_j)$

$\rightarrow F(D_i) < F(D_j)$

$S(D_i) = S(D_j)$

$\rightarrow \because Dis(D_i) > Dis(D_j)$

$\rightarrow F(D_i) < F(D_j)$

$\rightarrow r \geqq m, D_i \in DO, F(D_i) < F(D_j)$ □

From theorem 2, because the scores of normal nodes are bigger than those of outliers and we will filter the datasets whose scores are smaller, the outliers would be filtered before the normal datasets. As the proof of these theorems, we demonstrate that the propagation method in the phase I can provide strong protection of the spoofing attack. The security of phase II we demonstrate is that when the outlier ratio *r* is bigger than the ratio of normal datasets m, the outlier can be filtered successfully.

5. Evaluation

In the following section, we insert two types of outliers into our datasets separately. The relation of outliers in the first type is beyond the limit of physical crystals. Crystal's vendors could offer the maximum variance of skew like 40us per second. If the relation between the outlier and a normal dataset is outside this bound, we consider this outlier as extreme outlier. The other type of outlier is introduced by cooperative compromised nodes and their relation is inside the bound of physical limitation. We regard these outliers as mild outliers. Our simulation shows the end result after analyzing the different datasets. It uses MATLAB to generate the normal datasets and insert some outliers. Then we analyze the datasets and filter some outliers. Finally, we plot the error which is the difference between the calculated result and assumptive value at different cases.

5.1.Simulation setup

We assume a node received 32 datasets from its neighbors. The interval of the reception of each dataset is more than 500ms. Our assumptive relation of clock skew between its local clock and global clock is 1.00004 and the initial clock offset is 6 second. In our simulation, we exercise 6 different filter ratio and 6 different outlier ratio, separately 0, 0.1, 0.2, 0.3, 0.4 and 0.5.

In each case, we run 32 rounds to get the averages of calculated clock skew and initial offset. The error is the difference between the average and assumptive value. We will plot the relation between error, filter ratio and outlier ratio.

5.2. Extreme Outliers

The first scenario we use to analyze is the datasets which contain the extreme outliers. Because the extreme outliers do not cooperate, their relations are not inside the bound of physical limitation. So these extreme outliers would be filtered at score phase easily.

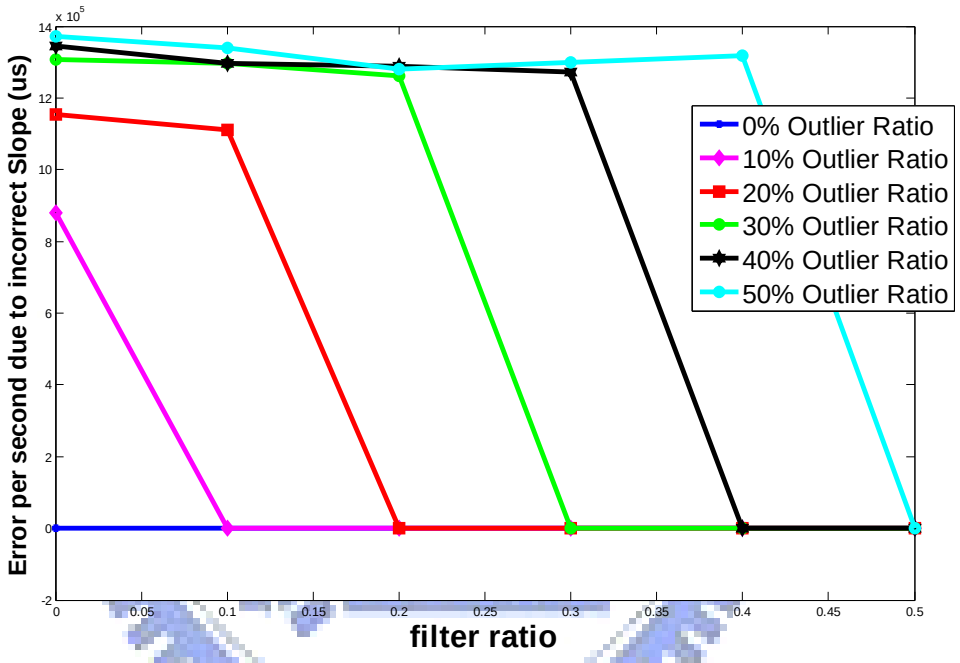


Figure 5-1: The average error between calculated slope with extreme outliers and the assumptive slope in each case

We could see the result from Figure 5-1 that when the filter ratio is more than the outlier ratio, the average error is smaller than 1us. This result is like the result of FTSP without attack and it represents that we filter the outliers specifically. There is strangeness in Figure 5-1. When the outlier ratio becomes bigger, the error is not more acute. The reason is that the scale of figure is too large relative to the difference of error. Actually the error increases about 0.003 (3ms) while the outlier ratio increases

0.1. The error is not absolute and is relative to the distribution of the attacked datasets. After our filter phase, the average error being smaller than 1us means that a node don't need to resynchronize time too frequently and could consume as less power as possible.

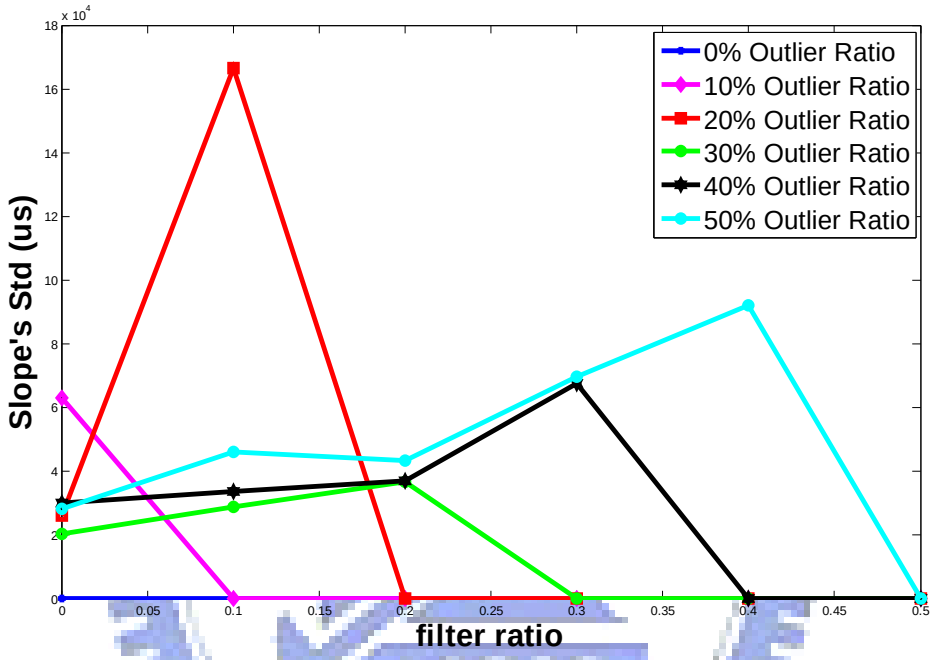


Figure 5-2: The standard deviation of the calculated slopes with extreme outliers in each case

Figure 5-2 shows the standard deviation of slopes in each case. We observed that if there are outliers in datasets, the standard deviation of slope is very large. This is very critical for time synchronization, because the difference between each node's skews vary very much. Hence, each node's clock varies vastly. Another observation is that if we filter some part of outliers in such a case, the standard deviation becomes bigger. So we could infer that when there are few outliers in datasets, it could result in serious error. Fortunately, if the filter ratio is more than or equal to the outlier ratio, the standard deviation would downgrade to about the order of microsecond.

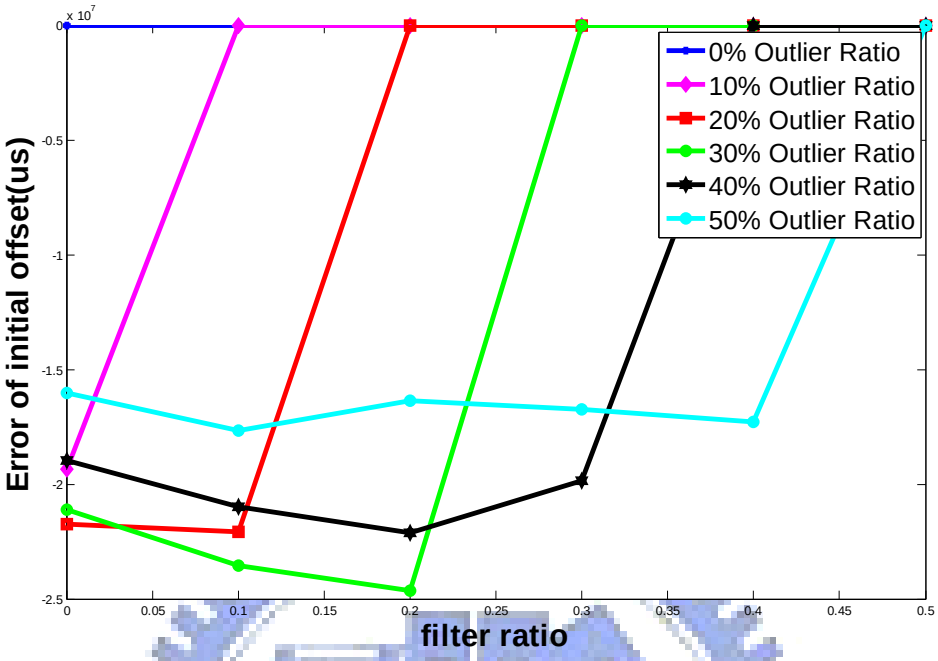


Figure 5-3: The average error between calculated offset with extreme outliers and the assumptive offset in each case

The average error between calculated offset and the assumptive offset in each case is showed as Figure 5-3. Because of the existence of outliers, the severe error could be up to the ten orders of second. The error of offset, like the error of skew, is not absolute and it is relative to the distribution of the attacked datasets. This error could be reduced to the ten orders of microsecond by increasing the filter ratio. Although initial offset is a fixed value and the node does not use it to compensate the clock drift, we could note that the offset error is much bigger than the skew error. This critical error could influence many applications and induces some serious error.

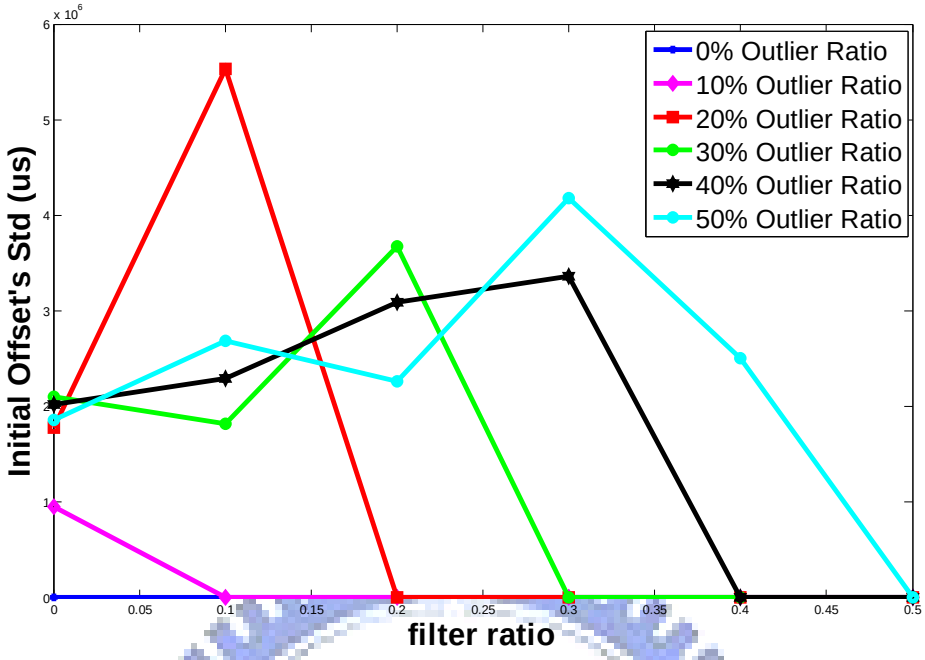


Figure 5-4: The standard deviation of the calculated offsets with extreme outliers in each case

The last evaluation with the existence of extreme outliers is the standard deviation of the initial offsets. Figure 5-4 shows the standard deviation of the initial offset is also huge like that of slope. This represents if there are a few outliers in datasets, the initial offset vary very much.

From above figures, we observe some strangeness that how we could promise to filter the half of datasets which are just outliers if the ratio of outliers is 0.5. The cause is that these outliers are not mild outliers and their scores are certainly smaller than those of normal datasets because they don't cooperate with each other. If the outliers are all mild outliers and the attacked ratio is more than 0.5, we cannot filter them effectively. We would examine this case at next simulation.

5.3. Mild Outliers

The second scenario we want to analyze is the datasets which contain the mild

outliers. The mild outliers are introduced by cooperative compromised nodes and their relations are inside the bound of physical limitation. The mild outliers could not be filtered at score filter but could be at regressive filter. The difference between mild outliers and extreme outliers is if the attackers cooperate with each other or not. The key to filtering the outliers successfully is that the amount of outliers must be less than that of normal datasets. So the regression line would be near the most datasets. From underlying figures, we could discover the truth we cannot work successfully when the outlier ratio is equal to or more than the filter ratio.

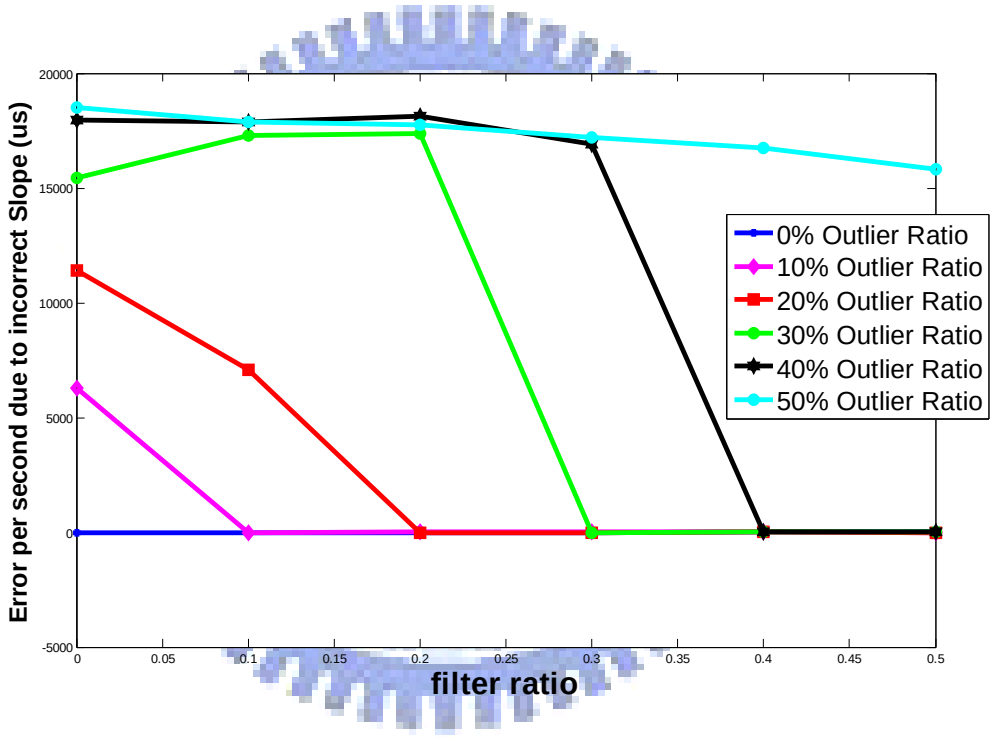


Figure 5-5: The average error between calculated slope with mild outliers and the assumptive slope in each case

In Figure 5-5, the average error of the datasets including mild outliers is smaller than those including extreme outliers. The importance of that is when the outlier ratio is equal to the filter ratio; both of our two filtering methods cannot work productively. The outliers' scores are equal to the normal datasets, so we could not filter any outlier at score filter. In the regressive filter, the regression may tend towards the incorrect

trend. So we are likely to filter some normal datasets and then calculated wrong answer.

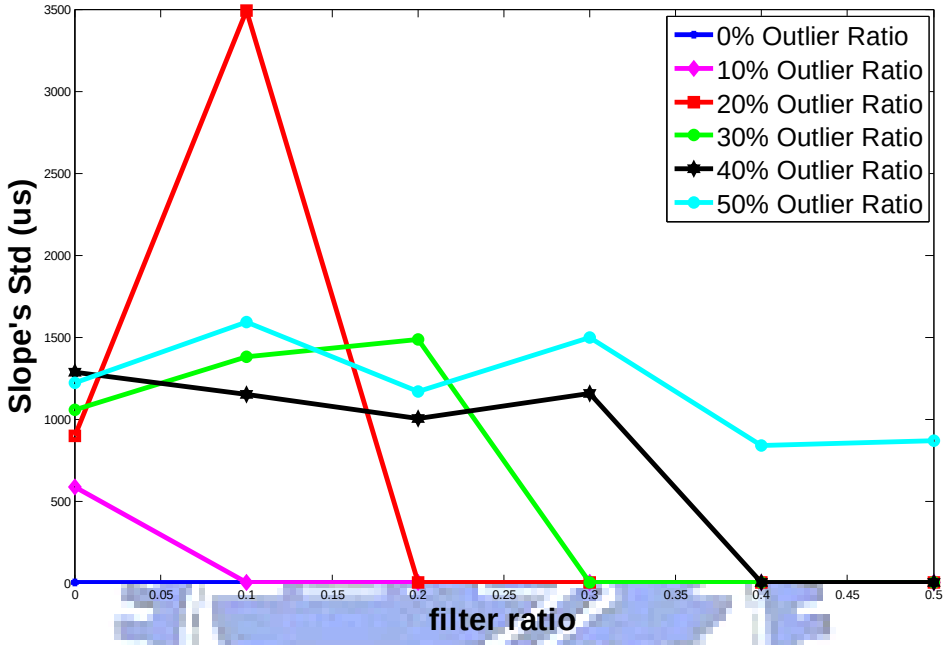


Figure 5-6: The standard deviation of the calculated slopes with mild outliers in each case

Figure 5-6 shows the standard deviation of slopes in each case. The result is like Figure 5-2. If there are few outliers, the computed output is unbelievable and the standard deviation is unpredictable. The difference from Figure 5-2 is that when the filter ratio and outlier ratio both are half of datasets, our result is not ideal because we cannot filter the outliers specifically.

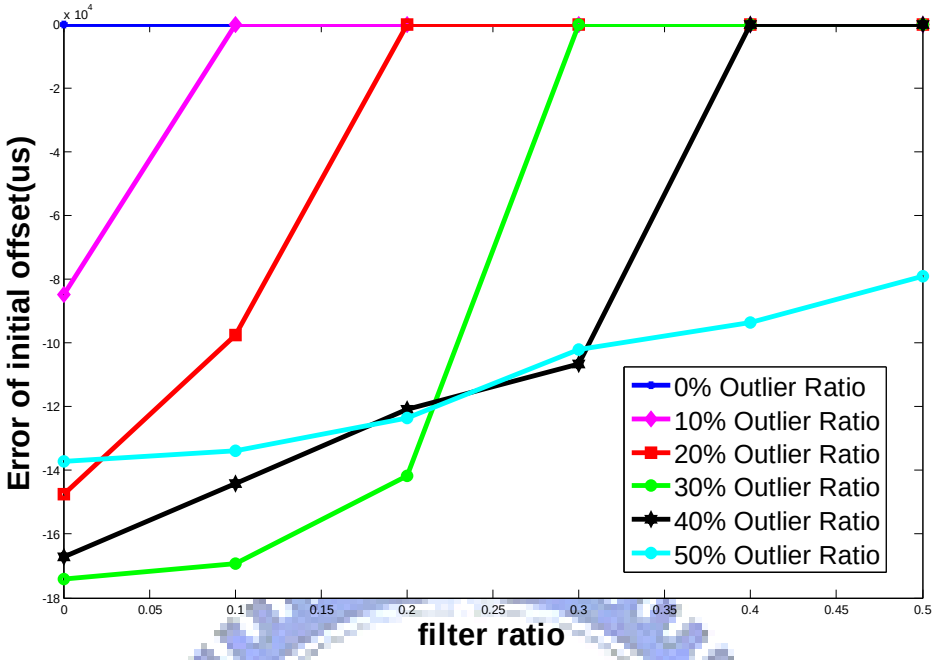


Figure 5-7: The average error between calculated offset with mild outliers and the assumptive offset in each case

The average error between calculated offset with mild outliers and the assumptive offset in each case is showed as Figure 5-7. Because the outliers are mild, the average error is about a few seconds. Although the error is smaller than that of extreme outliers, we still cannot tolerate it. In Figure 5-7, we could see that the error was reduced to the ten orders of microsecond by letting the filter ratio be more than the outlier ratio.

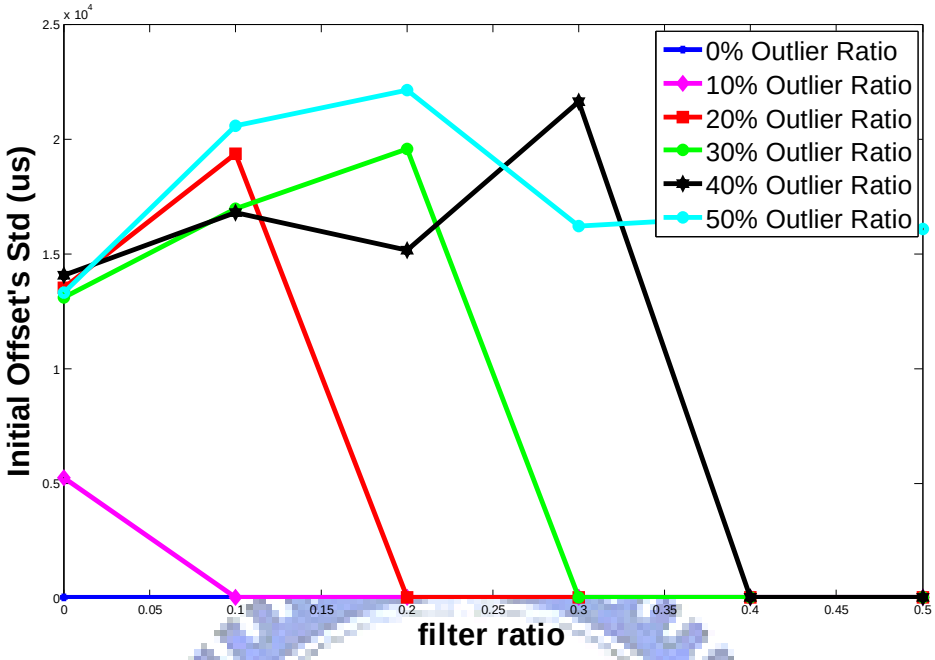


Figure 5-8: The standard deviation of the calculated offsets with mild outliers in each case

Figure 5-8 shows that the standard deviation of initial offsets is also huge like that of slope. It also supports the same truth that if the outliers are all mild outliers and the outlier ratio is more than half, we cannot filter them effectively.

6. Conclusion

In this paper, we proposed a time synchronization protocol based on FTSP, called Outlier-Filtered Time Synchronization Protocol for WSNs. It points the insufficiencies of FTSP and could resist attacks. FTSP deals with neither external attackers nor insider attackers. We proposed three methods for promising the authenticity and integrity of the messages to defend against external attackers. The first method is that communicating through unicast with each neighbor. The second method is to broadcast the message with multiple MACs. The last method is an *u*TESLA-based approach with a little modification. Their functionalities are all the same for avoiding external attackers forging and spoofing messages, but the limitations and overheads are different. Users could depend on environment to select a suitable one.

To defend against insider attack and pulse-delay attack, we design a neat method due to the resource constraints of the sensors. The filter ratio is an adjustable parameter. The filter ratio could be set by the estimated ratio of outliers and also could be automatically tuned by extra computation. The smaller filter ratio means that we retain more reference points and the predictive relation between local clock and global clock will be the more precise. But smaller filter ratio means the security level is lower.

Based on our experiments, we inserted both extreme and mild outliers into the normal datasets and observed that RTSP keeps the same precision as that of FTSP even when some attacks take place. The average error of skew could be less than 1us and the average error of offset could be less than 20us. The lower error of clock skew means that we don’t need to resynchronize too frequently, so we could save more

energy.

References

[1] Mills, D. L., “Internet Time Synchronization: The Network Time Protocol. Global States and Time in Distributed Systems,” IEEE Computer Society Press, 1994.

[2] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins “GPS Theory and Practice,” SpringerWienNewYork, 1997.

[3] Donggang Liu, Peng Ning, Rongfang Li, “Establishing Pairwise Keys in Distributed Sensor Networks,” ACM Transactions on Information and System Security, February 2005.

[4] W. Du, J. Deng, Y. S. Han, and P. Varshney, “A Pairwise Key Predistribution Scheme for Wireless Sensor Networks,” Information and System Security, October 2003.

[5] J. Elson, L. Girod, and D. Estrin, “Fine-Grained Network Time Synchronization Using Reference Broadcasts,” ACM SIGOPS Operating System, 2002.

[6] Sichitiu, M.L., Veerarittiphan, C., ” Simple, Accurate Time Synchronization for Wireless Sensor Networks,” Wireless Communications and Networking IEEE, March 2003.

[7] Ganeriwal, S., Kumar, R., and Srivastava, M. B., “Timing-Sync Protocol for Sensor Networks,” Embedded Networked Sensor System, November 2003.

[8] Y. Hu, A. Perrig, and D. Johnson, “Packet Leashes: A Defense Against Wormhole Attacks in Wireless Ad Hoc Networks,” INFOCOM, April 2003.

[9] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, “The flooding time synchronization protocol,” in Proc. 2nd ACM Conf. Embedded Networked

Sensor Syst., Nov. 2004, pp. 39–49.

[10] Kopetz, H., and Ochsenreiter, W. Clock Synchronization in Distributed Real-Time Systems. IEEE Transactions on Computers, C-36(8), p. 933–939, August 1987.

[11] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, USA, September 2002.

[12] C. Lin, C. Federspiel, and D. Auslander, "Multi-Sensor Single Actuator Control of HVAC Systems," International Conference for Enhanced Building Operations, 2002.

[13] N. Xu, "A Survey of Sensor Network Applications," http://enl.usc.edu/ningxu/papers, 2004.

[14] L. Schwiebert, S. Gupta, and J.Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," Mobile Computing and Networking, 2001.

[15] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care," Wearable and Implantable Body Sensor Networks, 2004.

[16] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, August, 2002.

[17] S. Slijepcevic, M. Potkonjak,V. Tsiatsis, S. Zimbeck, M. B. Srivastava, "On Communication Security in Wireless Ad-Hoc Sensor Network," Infrastructure for Collaborative Enterprises, June 2002.

[18] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," Mobile Computing and Networking, August 1999.

[19] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, “A Taxonomy of Wireless Microsensor Network Models,” ACM Mobile Computing and Communications , 2002.

[20] Saurabh Ganeriwal, Srdjan Capkun, Chih-Chieh Han, Mani B. Srivastava, “Secure Time Synchronization Service for Sensor Networks,” Mobile Computing and Networking, September 2, 2005

[21] M. Manzo, T. Roosta, and S. Sastry., “Time Synchronization Attacks in Sensor Networks,” Security of Ad Hoc and Sensor Networks, 2005.

[22] H. Song, S. Zhu, and G. Cao., “Attack-Resilient Time Synchronization for Wireless Sensor Networks,” Mobile Ad-hoc and Sensor Systems, 2005.

[23] K. Sun, P. Ning, and C. Wang., “Secure and Resilient Clock Synchronization in Wireless Sensor Networks,” IEEE Journal on Selected Areas in Communications, February 2006.

[24] J. R. Douceur., “The Sybil Attack,” Peer-to-Peer Systems, March 2002.

[25] J. Newsome, R. Shi, D. Song, and A. Perrig., “The Sybil Attack in Sensor Networks: Analysis and Defenses,” Information Processing in Sensor Networks, April 2004.

[26] J. van Greunen and J. Rabaey., “Lightweight Time Synchronization for Sensor Networks,” Wireless Sensor Networks and Applications, September 2003

[27] H. Dai and R. Han., “TSync : A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks,” ACM Sigmobile Mobile Computing and Communications, January 2004.

[28] Mingxia Xu, Minjian Zhao, Shiju Li., “Lightweight and Energy Efficient Time Synchronization for Sensor Network,” Wireless Communications, Networking and Mobile Computing, September 2005.

[29] D. L. Mills. W. Su and I. F. Akyildiz., “Time-Diffusion Synchronization Protocol

for Wireless Sensor Networks," IEEE/ACM Transactions On Networking, April 2005.

[30] Santashil PalChaudhuri. Amit Kumar Saha. David B. Johnson., "Adaptive Clock Synchronization in Sensor Networks," Information Processing In Sensor Networks, 2004.

[31] Kun Sun, Peng Ning. Cliff Wang. An Liu, Yuzheng Zhou., "TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks," Computer and Communications Security, October, 2006

[32] Elson, J., Romer, K., "Wireless Sensor Networks: A New Regime for Time Synchronization," Hot Topics in Networks, Princeton, New Jersey, 2002

[33] http://en.wikipedia.org/wiki/Outlier