A Security Analysis of the Precise Time Protocol

Jeanette Tsang and Konstantin Beznosov

Laboratory for Education and Research in Secure Systems Engineering
lersse.ece.ubc.ca
University of British Columbia
Vancouver, Canada

Technical report LERSSE-TR-2006-02*

Last Modification Date: 2006/12/04

Revision: #6

*This and other LERSSE publications can be found at http://lersse-dl.ece.ubc.ca

Abstract

This paper reports on a security analysis of the IEEE 1588 standard, a.k.a. Precise Time Protocol (PTP). We show that attackers can use the protocol to (a) incorrectly resynchronize clocks, (b) illegally rearrange or disrupt the hierarchy of PTP clocks, (c) bring the protocol participants into an inconsistent state, or (d) deprive victim slave clocks from synchronization in ways undetectable by generic network intrusion detection systems. We also propose countermeasures for the identified attacks.

Keywords: IEEE 1588, Precise Time Protocol, Network Time Protocol, security analysis.

Contents

1 Introduction

The ability to precisely synchronize clocks among distributed components is critical for electrical power systems, industrial automation, telecommunication systems, military applications, and other fields where timing is crucial to their correctness and performance. The components of these distributed systems often contain real-time clocks that control their performance and coordination. Three types of time-dependent controls commonly used in automation are message-based, periodic, and time-based. An example of message-based control in a real-time system would be to generate a periodic interrupt by matching the time from the clock [22]. Checking the value of a sensor at fixed periods of time is an example of a periodic control. Waking up a system component at 5:00 AM would be an example of time-based control. Other examples of the significant role played by accuracy of time in measurement and control systems can be found in [7].

The IEEE 1588 standard [1] specifies a precision clock synchronization protocol for networked measurement and control systems that may utilize non-IP networks. It is equivalent to the IEC 61588 standard [14]. Both standards are known as Precise Time Protocol (PTP). In this paper, we use the terms PTP and IEEE 1588 interchangeably. Networked heterogeneous systems can employ this protocol to synchronize clocks with accuracy in the submicrosecond range. The protocol uses the multicasting messaging property of the network (e.g., Ethernet) to exchange messages between master and slave nodes – commonly referred to as "master clocks" and "slave clocks" to obtain synchronization. An example of real-world usage of IEEE 1588 applications in a distributed motion control system is presented in [18]. An example of its application in power line networks is provided in [19].

Although existing protocols such as the Network Time Protocol (NTP) [15] and the Global Positioning System (GPS) are used to synchronize clocks within the network, PTP is the only one that offers accuracy at the sub-microsecond level for small self-administered networks [8, 11]. NTP also requires underlying network to be IP-based, whereas PTP does not have this restriction.

Even though PTP is being positioned by the industry to serve as a key time synchronization technology for automation and control [19], its resilience to security attacks has not yet been publicly studied. For PTP to serve its intended role, the automation and control community needs to be aware of the protocols security properties.

Related works on time synchronization protocols for wireless sensor networks [23, 24] also suggest existence of security issues such as modification and forgery of synchronization messages, and being vulnerable to denial of service and delay attacks. Few of the proposed countermeasures include the use of secure pairwise synchronization to authenticate nodes locally, then extending it to a group-wise, or even a global time synchronization. But these methods would not work in a PTP network, because it will require changes to the PTP's fundamental message exchanging mechanism. Yet, other suggested methods such as to impose additional authentication mechanisms on (broadcast) messages to ensure the authenticity of the source, content and timeliness (e.g., the μTESLA protocol [25]), and using message integrity code (MIC) to check unintended modification of the messages show could probably be useful as part of countermeasures to the attacks mentioned in later part of this report.

In this paper, we report on the results of our security analysis of the 2002 revision of the IEEE 1588 specification [1]. The analysis focused on PTPs message transmission period,

i.e., when networked devices exchange synchronization messages. The results of this analysis can assist the developers and users of PTP-based technologies in identifying the security requirements and developing the necessary security mechanisms for the protocol.

We made several assumptions for the purpose of analysis. In order to focus on PTP-related attacks and to exclude attacks specific to other protocols from our analysis, we assumed that the network being analyzed is a closed network, which means that none of the network nodes is connected to other external networks, such as the Internet. For those environments where the above assumption does not hold, additional types of attacks (e.g., distributed denial of service), including those specific to general-purpose network protocols (e.g., IP, UDP, TCP), have to be taken into account. As a consequence of the first assumption, our second assumption was that only insiders, i.e., adversaries who have direct access to the PTP network, can initiate attacks. Our third assumption was that the attacker(s) can mount passive (message eavesdropping) as well as active (message modification, removal, and injection) attacks. On the other hand, we could not make the assumption that IPSec [17] and its supporting services (e.g., key management) are available in the automation and control system that uses PTP because the PTP specification does not mandate IPSec.

Due to the lack of built-in protection, PTP messages can be easily tampered with by anyone who has access to the network. More importantly, the results of our analysis also suggest that attackers can easily use this weakness to incorrectly resynchronize clocks or to illegally rearrange (or even disrupt) the hierarchy of PTP clocks. Additionally, the protocol lacks a mechanism for detecting and compensating out of range data that can result in an inconsistent state of PTP participants. Furthermore, we discovered several PTP-specific attacks that, we believe, are very hard to detect by an intrusion detection system unless it maintains the state of the victim PTP clock hierarchy, which is expensive.

The rest of the paper is organized as follows: an overview of the IEEE 1588 standard is given in Section 2, which describes the clock and message types used in the standard as well as key algorithms. Section 3 describes the attacks that PTP was identified to be vulnerable to. Section 4 provides a summary of the results obtained from our security analysis. We draw conclusions and outline future work in Section 5.

2 IEEE 1588 Standard

This description of the standard is based on the 2002 revision of the IEEE 1588 specification [1]. The main two elements of any PTP network are the *clock* and *port*. The *clock* is a network node that can provide measurements of time. The *port* is a logical access point to a clock. An example of a clock is a network switch connected to two or more subnets. Its connections with the subnets are referred to as *ports*. The switch is referred to as a *boundary clock*, which will be described further later.

The precision of the synchronization in a PTP network relies significantly on the delay fluctuations of the network and its components. For this reason, the protocol accommodates less deterministic components with notable delay fluctuation, e.g., routers and switches, by the use of *boundary clocks*, clocks with more than one port [11]. *Ordinary clocks* only have a single port.

Each clock port serves either as a master or slave. The standard refers to these roles as

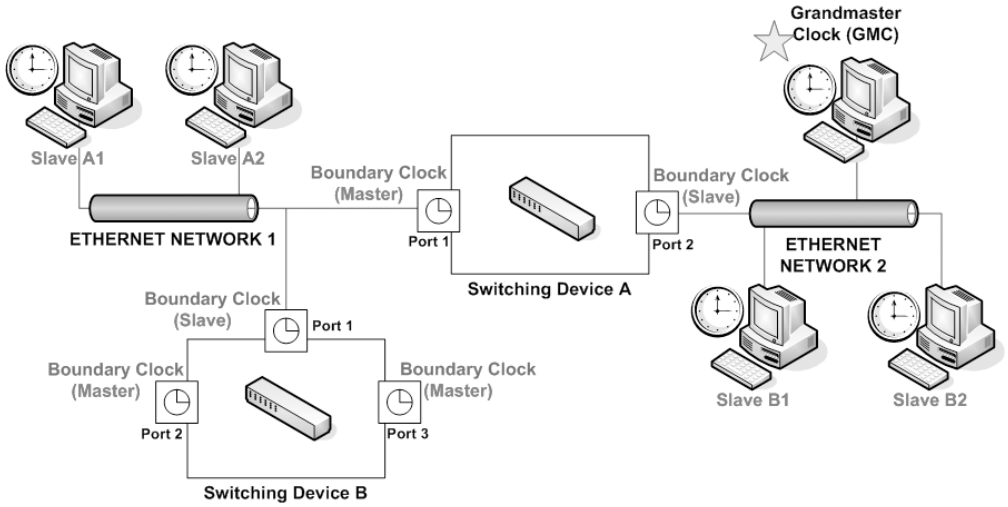


Figure 1: A hypothetical PTP network (adapted from [6]).

master clock or *slave clock*. The master clock is used as a reference for calibrating all of the slave clocks. The most precise clock in a PTP network serves as a master. It is elected via the best master clock (BMC) algorithm, which is executed by every port individually. The algorithm ensures that there is only one master clock active at any given moment on any subnet. The BMC algorithm is explained in detail in Section 2.3 and its vulnerabilities are analyzed in Section 3.1.

Figure 1 shows a hypothetical PTP network with two subnets connected via one switch. Switching Device A is a boundary clock with two ports. Port 2 acts as a slave, and port 1 acts as master clock. The standard allows messages to be transmitted either directly in Ethernet frames or as UDP payload.

All clocks in a PTP network are organized into a tree-like hierarchy according to their stratum (a.k.a. class) numbers. The clocks stratum number represents one measure of that clocks quality. The smaller the stratum number of a clock the better the clock is. The root clock has stratum number one and provides the reference time for other clocks in the network. A calibrated atomic clock or a global positioning system (GPS) receiver is often chosen as the root clock. Clocks with greater stratum number are generally further away from the root clock in the PTP hierarchy. Thus, a clock with a greater stratum number is less reliable for calibrating other clocks.

When multiple subnets are joined to form a bigger network, the master clocks of these subnets will be synchronized with the root clock, a.k.a. *grand master clock* (GMC). GMC is also determined using the BMC algorithm. There is only one GMC per PTP network, and one master clock per subnet. The hierarchical order of the clocks is as follows: GMC, master clocks (M), and then slave clocks (S). The top-right node in the figure is the GMC for the PTP network.

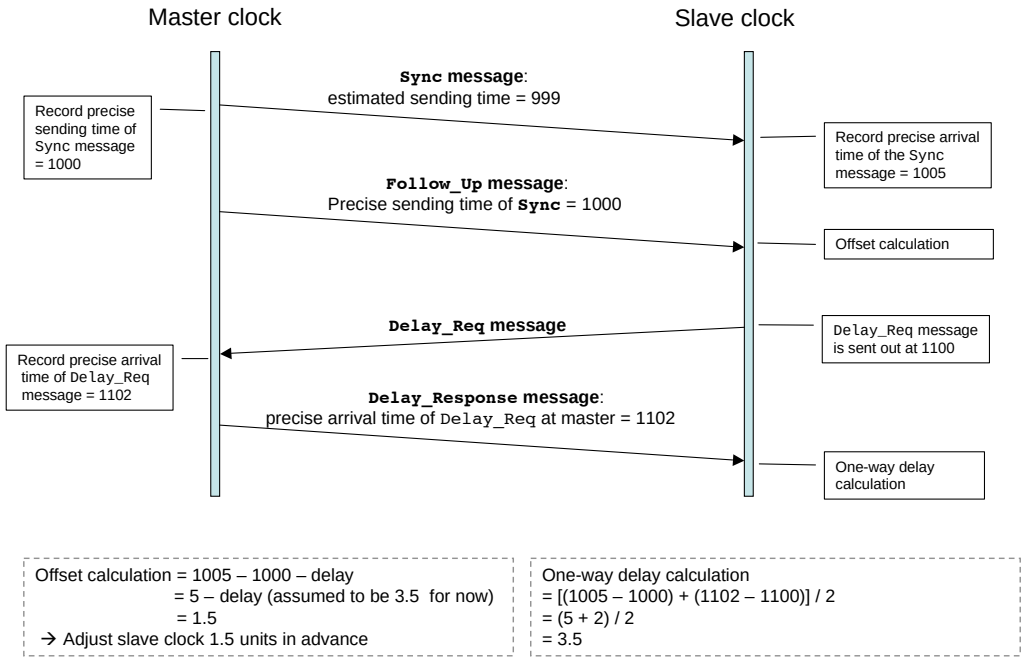


Figure 2: Time sequence diagram of messages exchanged between master and slave clock to achieve time synchronization

2.1 Time Synchronization

Time synchronization is performed over three phases: master clock selection, time offset correction, and communication delay measurement. The first two phases are executed every synchronization interval, which is 2 seconds by default [11]. Delay measurement is initiated by each slave individually on irregular bases, between 4 and 60 seconds by default [11].

In the following subsections, each phase is discussed in detail along with a simplified example of how time synchronization works in PTP. The message flow for the example is illustrated in Figure 2. This example is used again later in this paper.

2.1.1 Time Offset Correction

During this phase, each slave clock corrects its time using the time of the master clock. It obtains the masters time from `Sync` message. The master can also send an optional `Follow_Up` message to improve the accuracy of the correction.

Synchronization Message: The `Sync` message is multicasted by the master clock to all the slaves on the subnet. The main purpose of the message is to deliver the estimated time that it has left the master clock. Upon reception of this message, the slave clocks record the reception time. Each slave clock executes the following steps upon receiving a `Sync` message:

1. Check if the clock is in the slave mode. If not, go to step A below.
2. Check to see if the `Sync` message is sent from the current master. Otherwise go to step A.
3. Verify that the `sequenceID` of the `Sync` message is greater than the `parent_last_sync_sequence`, which is stored in the parent data set of the slave clock. Otherwise discard the message.
4. Update parent data set with the information from the message (e.g., `sequenceID`).
5. Check the `Sync` message to see if the master will send a `Follow_Up` message. If so, do nothing and wait for the `Follow_Up` message. If not, synchronize the local time with the time from the message taking into account the communication delay computed during communication delay measurement phase. Stop processing the message.

Step A: Update existing Foreign Master Data Set with the information about the master clock, and ignore the message.

Foreign Master Data Set is used when calculating the best master clock algorithm, which will be discussed in depth in Section 2.3.

Referring again to the example illustrated in Figure 2, let us assume that the master clock sent out a `Sync` message with *estimated sending time* = 999. Immediately after the transmission, the master clock records 1000 to be the *precise sending time*. The slave clock receives the message at 1005. The reception time is used after receiving the upcoming `Follow_Up` message, to calculate the offset between the slave and master clock. If no `Follow_Up` message is used, the slave clock immediately synchronizes to the time calculated by (1):

$$O = R_{Sync} - E_{Sync} - D \tag{1}$$

Where

- O is the slave clocks offset from the master clock
- R_{Sync} is the reception time of the `Sync` message
- E_{Sync} is the estimated sending time of the `Sync` message
- D is the one-way delay between master and slave clocks (for now it is assumed to be 3.5)

Follow Up message: The optional `Follow_Up` message is sent immediately following the `Sync` message by the master containing the precise sending time of the `Sync` message. After receiving the `Follow_Up` message, the slave clock calculates the offset by using (2):

$$O = R_{Sync} - P_{Sync} - D \tag{2}$$

Equations (3), (4), (5), and (6) are used to calculate the one-way delay from the master and slave clocks. As mentioned earlier, the delay is assumed to be symmetric; thus, an average is taken in 3. Equation (4) is used when a `Follow_Up` message is sent, whereas Equation (5) is used when a `Follow_Up` message is not sent.

$$D = (D_{ms} + D_{sm})/2 \tag{3}$$
$$D_{ms} = R_{Sync} - P_{Sync} \tag{4}$$
$$D_{ms} = R_{Sync} - E_{Sync} \tag{5}$$
$$D_{sm} = R_{Delay\_Req} - P_{Delay\_Req} \tag{6}$$

Where

- D_{ms} = master-to-slave delay
- D_{sm} = slave-to-master delay
- R_{Delay_Req} = reception time of the Delay_Req message
- P_{Delay_Req} = precise sending time of the Delay_Req message

The following procedure is used by the slave clocks to ensure that the `Follow_Up` message is valid.

1. Check to see if the `Follow_Up` message is from the current master. If not, then the `Follow_Up` message is discarded.
2. Ensure that the `associatedSequenceId` is equal to `parent_last_sync_sequence`, which is stored in the parent data set of the slave clock. If not, then the `Follow_Up` message is discarded.
3. Synchronize local clock in slave.

Referring back to the example in Figure 2, in the `Follow_Up` message received by the slave clock, the precise sending time of the `Sync` message is 1000. The one-way delay is 3.5 time units, which is calculated from the `Delay_Req and Delay_Response` messages. The offset is 1.5 time units. The local clock of the slave is then recalibrated using the calculated offset value.

2.1.2 Computing Communication Delay

Communication delay between each slave and the master clock is computed with the aid of delay request and response messages.

Delay Request message: The `Delay_Req` message is initiated by the slave clock, and is sent periodically to the master at irregular intervals. Its main purpose is to find out the communication delay between the slave and master clocks, as stated in equation 5. Upon reception of this message, the master clock records its exact arrival time. In the Figure 2 example, the `Delay_Req` message is sent to the master clock at precisely 1100, and the master clock receives the message at 1102.

Delay Response message: The master clock responds to a `Delay_Req` message by sending back a `Delay_Response` message. This message contains the precise reception time of the `Delay_Req` message, 1102 in our example. Thus, using equation 5, the slave-to-master delay is calculated to be of 2 time units. Combining the master-to-slave delay, which is 5 time units, and dividing it by one-half, the one-way communication delay between master and slave is found to be 3.5 time units.

2.2 Message Format

The standard allows messages to be transmitted either directly in Ethernet frames or as UDP payload. All PTP messages can be divided into two major groups: management messages and time synchronization messages. In our paper, we concentrate only on the latter.

There are four types of synchronization messages, as noted in the previous section: `Sync`, `Follow_Up`, `Delay_Req`, and `Delay_Response`. They are used for the regular synchronization procedure, and are propagated only within one subnet. The contents of the `Sync` and `Delay_Req` messages are listed in Appendix B. Those fields listed in Appendix B that are also contained in the `Follow_Up` and `Delay_Response` messages are indicated with asterisks.

2.3 Best Master Clock Algorithm

The best master clock (BMC) algorithm is used for determining which PTP clock is closer to the grandmaster clock. The algorithm is run autonomously at each port of every PTP clock at least once every sync interval. The result of running the algorithm determines the mode of the clocks port for the next sync interval. The algorithm computation is based on the following: information from the most recent `Sync` messages, the clocks default data set, and the foreign master data set.

The BMC algorithm has two parts: state decision and data set comparison. For the state decision part, a clock's default data set (DDS) is being compared with `Sync` message(s). Information used for comparison includes various fields in the DDS, for example the information related to the grandmaster clock. If the local clock running the BMC algorithm is a very high quality clock, (e.g., a calibrated atomic clock, a clock designated as a reference clock), it follows the path shown on the left side of Figure 3. The clocks DDS is compared with the best data set of all qualified `Sync` messages received by other ports connected to the same communication path as the port running the BMC. We will refer to this set as $\texttt{Sync}_C$. It is essential for our security analysis to understand that the purpose of comparison is to determine which clock is derived from the better GMC rather than which is a better clock. If the DDS wins the comparison, then it means the local clock is derived from a better GMC. As a result, the local clock would be in the master mode until another clock wins. However, if it loses, the local clock would switch into *passive* mode, in which it does not send any messages to other clocks.[1]

Alternatively, if the local clock is of lower quality (i.e., its stratum number is greater than 2), it follows the path shown on the right side of Figure 3. Its DDS is compared with

[1]The key difference between slave and passive states of clocks are that the port associated with a clock in slave mode does not send any messages on its communication path.

the best of all $\texttt{Sync}_C$ messages from all ports of the clock, referred to collectively as $\texttt{Sync}_A$. Again, if the DDS is better, the local clock would run in master mode. Yet, if $\texttt{Sync}_A$ wins, the comparison carries on to weighing $\texttt{Sync}_A$ and $\texttt{Sync}_C$. If both are the same, then the local clock would be in slave mode, for some other port connected to the local clock is better than its own. On the other hand, if $\texttt{Sync}_A$ is better, then it would be in passive mode. It would proceed on as master mode if $\texttt{Sync}_C$ is better than $\texttt{Sync}_A$, because this means that other ports of the local clock are derived from a better GMC than all the clocks connected to the communication path of this particular port running the BMC algorithm (as represented by $\texttt{Sync}_C$).

The data set comparison algorithm is used within the state decision algorithm to define the various comparisons for determining which of DDS, $\texttt{Sync}_C$ or $\texttt{Sync}_A$ is better. It uses parameters such as: `grandmasterClockUuid`, `grandmasterClockStratum`, `grandmasterClockIdentifier`, and `grandmasterClockVariance`, for judging. Discussion of how the algorithm works is beyond the scope of this paper. Diagrams on pages 68-71 of [1] depict the full flow chart of this data set comparison. It is important to note that the fields in a `Sync` message are used locally by each PTP clock in computing BMC to determine the mode of the clock. The reliance exclusively on the information from `Sync` messages for computing BMC and the lack of coordination among clocks during the election makes PTP participants vulnerable to protocol-specific attacks that are simple, but difficult to detect with generic IDS techniques.

3 Threat Analysis

Like Bishop's security analysis of the NTPv2 [4], we organized the discussion of each examined attack according to the attack goal(s), means, effects, and possible countermeasures for the following five types of threats: modification, masquerading, delay, replay, and denial of service. Specifically, we studied how these attacks could jeopardize synchronization objectives of PTP participants. The following subsections discuss results of our analysis for each of the five threat types.

3.1 Modification

Goals: Any of the following: (a) Cause denial of service (b) Cause slave clocks(s) to incorrectly resynchronize (c) Alter the hierarchy of the master and slave clocks

Attack: Manipulate the content of the message.

Effects: Modification of the messages sent by a master clock would produce the greatest effect, since a master clock can send messages used for both time synchronization and management.

(a) Attacking for denial of service

In the current version of the PTP, there is no mechanism for checking the authenticity of a message other than by checking the source of the message against the nodes data sets.

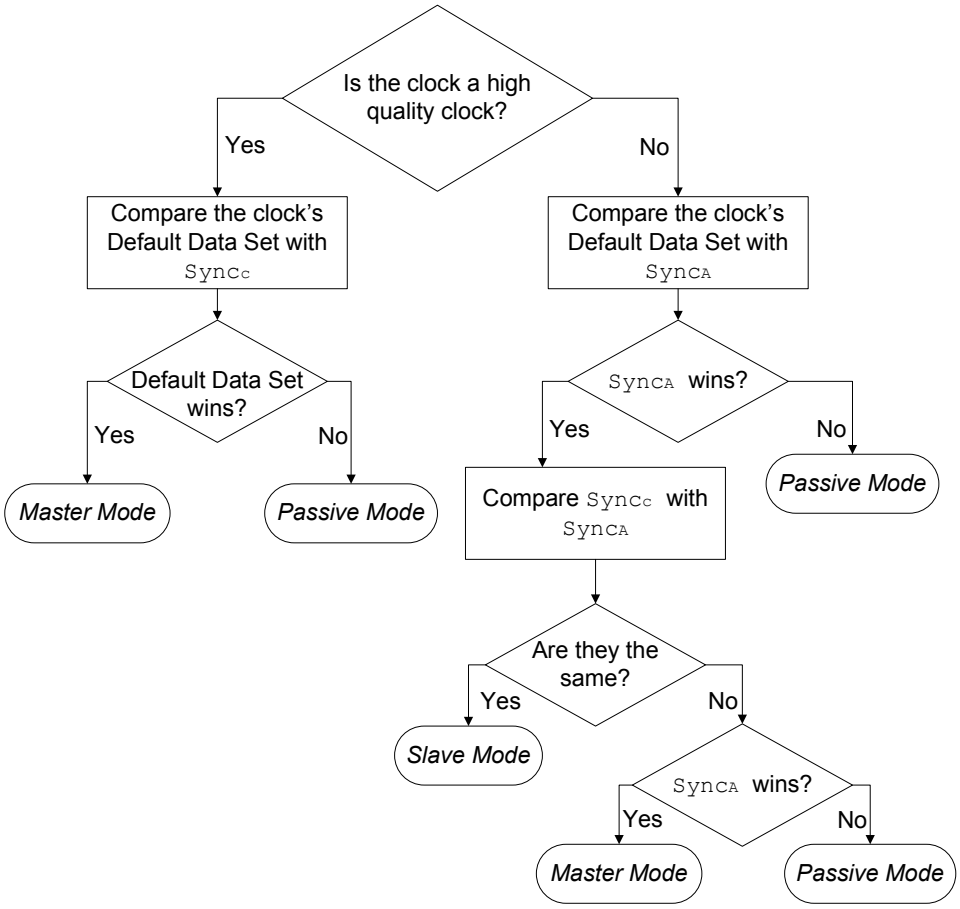


Figure 3: Flow chart of the BMC algorithm (from [1]).

The data sets contain information such as the local clock and parent clock attributes, and information about the current master, i.e., the clock whose `Sync` messages are used for correcting time. Slave clocks verify that a message came from the correct master by comparing the `sourceCommunicationTechnology`, `sourceUuid` and `sourcePortId` of the message (from Appendix B) with the `parent_communication_technology`, `parent_uuid_field` and `parent_port_id_field` in the parent data set of the slave clock. If the comparison fails, the message is discarded. By changing the above fields of the `Sync` messages, an attacker can make the matching of `Sync` messages fail; thus, slave clocks would refuse to synchronize with the true current master. This can cause a denial of service (DoS) attack without a generic intrusion detection system (IDS) or network sensors detecting the attack, unless the IDS knows the correct values of these fields. Furthermore, modifying the sequenceID value of the message can also lead to a PTP-specific DoS attack variant, which will be discussed in detail in Section 3.5.

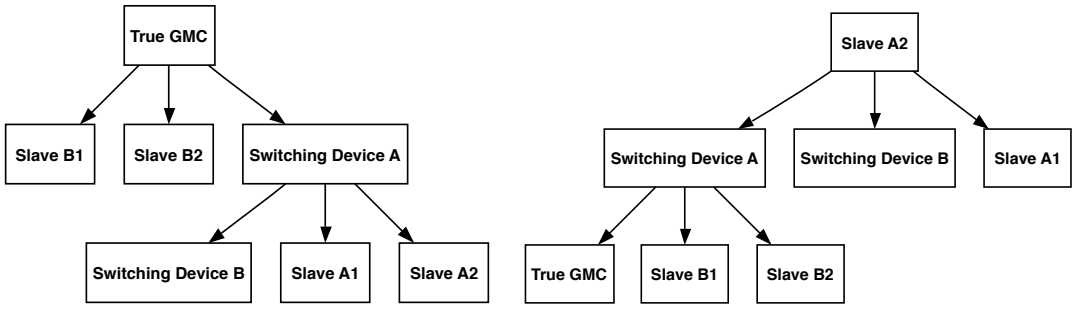
(b) Attacking for incorrect resynchronization

Tampering within the timestamp fields, such as `originTimestamp` and clock variance, of `Sync` messages can cause an incorrect resynchronization of the slave clock(s) or a miscalculation of the network latency. The `originTimestamp` field serves as the record of time at which the `Sync` message leaves the master clock. Modification of the time messages can be achieved by blocking the transmission of the original message, and subsequently injecting the modified message back into the communication channel.

(c) Attacking to alter the hierarchy of the master and slave clocks

Wrong information about the grandmaster clock within `Sync` messages can lead to setting the port to a different mode, e.g., slave or passive. A slave clock executes BMC algorithm for electing the best master clock for the next round of synchronization. As described in the previous section, the BMC algorithm uses information about the grandmaster clock (contained within a `Sync` message and DDS). By altering the grandmaster clock information in a `Sync` message, an attacker can easily make this message better than $\texttt{Sync}_C$ or $\texttt{Sync}_A$ of most clocks in the subnet. If the attacker performs further calculations based on `Sync` messages observed on the subnet, it can tailor the `Sync` message to be better than all the DDS and other `Sync` messages within the network. As a result, this crafted `Sync` message could become the $\texttt{Sync}_C$ (if the victim is a high quality clock) or $\texttt{Sync}_A$ (if the victim is not a high quality clock) to all local clocks connected to the attackers clock. Then by winning all the comparisons shown in Figure 3, the attacker can make the victim clock(s) switch into passive mode (for the left branch of the flow chart in Figure 3) or slave mode (for the right branch of the flow chart). As a result, the attacker could disrupt or even destroy the synchronization hierarchy of clocks on the victim PTP network.

To illustrate the above attack, consider the PTP network depicted in Figure 1. If, say, the attacker controls Slave A_2, it can start sending `Sync` messages that are better than those sent by the true grand master clock (GMC). As a result, Slave A_1 as well as switching devices A and B will elect Slave A_2 as their new master clock. Switching device A will also change its port 1 into slave mode and port 2 into master mode. As a result, the true GMC



(a) Before the attack (b) After the attack

Figure 4: PTP clock hierarchy before and after the hypothetical attack on the PTP network shown in Figure 1.

will switch into slave mode, and the original hierarchy of PTP clocks shown in Figure 4(a), will transform into the hierarchy shown in Figure reffig:afterAttack with Slave A_2 being a rogue GMC. Finally, modification attacks can be combined with other attacks, such as masquerading, which we describe in the following section.

Suggested countermeasures: We recommend employing cryptographic integrity protection on all PTP messages as a basic countermeasure. However, it is not obvious how the issue of key management can be addressed in PTP networks, as the protocol does not have any provisions for registering slave or master clocks, or binding their identities to keys. Efficiency is another issue to be considered, when cryptographic integrity protection is employed. Further investigation is needed to decide which cryptographic integrity protection is most suitable for PTP. An alternative countermeasure is to employ port-level security [20] to enforce such simple rules as only the network interfaces of the true GMC, port 1 of the switching device A, as well as ports 2 and 3 of the switching device B can send `Sync` messages. These access controls based on port-level security might be more effective performance-wise than those based on cryptography. In addition, key management can be avoided. However, the use of port-level security might fall short of enforcing all rules necessary to counter attacks on PTP clock hierarchies. For instance, if switching device A is compromised, then its port 1 can send bogus `Sync` messages even with the above rule enforced. It also increases the overhead of configuring network switches, and, as a result, increases the risk of friendly denial of service due to configuration errors.

3.2 Masquerading

Goals: To masquerade as the master clock, then use the false identity as a means of launching other attacks.

Attack: To obtain information about the "true" master clock, the attacker can eavesdrop on `Sync`, `Delay_Req`, and `Delay_Resp` messages that are sent to the slave clocks from the

master. The following fields used for verification of the master clock can be collected from the contents of the above messages: `sourceCommunicationTechnology`, `sourceUuid`, and `sourcePortId`. Once these data are obtained, the attacker can spoof `Sync`, `Delay_Req`, and `Delay_Resp` messages to masquerade as a master clock.

Effects: Camouflaging as a master clock could permit the attacker to send out incorrect timing and management messages to other slave clocks, causing different kinds of damage to the system. For example, the attacker can send out incorrect timing information to slave clocks, leading to errors in the synchronization process.

Suggested countermeasures: We recommend using a centralized or chained authentication process. For the centralized authentication, the grandmaster clock can act as the base of authentication management and, in addition, store any related information. For the chained authentication process, the authentication information of the new clock is passed on to the existing PTP network for verification via a network component that is already connected to the PTP network. This network component has previously authenticated the new clock by its own means.

As with the modification attack in Section 3.1, port-level security can be used to control which network device can send `Sync`, `Delay_Req`, and `Delay_Resp`. However, such controls reduce the robustness of the PTP in the presence of network failures. Since PTP clocks locally elect best master clocks for time synchronization, the role of master clock can be passed from clock to clock as the network topology changes due to device additions and failures.

3.3 Delay

Goal: Delay the arrival of messages at the recipient nodes, thus causing an increase in the values used in the offset and one-way delay calculations (equations (1) to (5)).

Attack: Through the use of hardware or software to interrupt the message transmission between nodes, and to later re-inject the message into the communication channel.

Effects: By intentionally delaying the reception time of the `Sync` message by a certain slave clock, the attacker may dramatically increase the offset of the slave clock with respect to the master clock, setting the slave clock off synchronization with the rest of the system.

Delaying the reception of the `Follow_Up` message at the slave clock can cause a timeout of the synchronization event (i.e., Phase 1 of the synchronization process). If this condition continues, it may lead to the slave clock being denied synchronization with the master. The slave will either pick the wrong clock on the subnet to synchronize with, or operate based on its local clock, eventually drifting from the true master clock.

A delay caused in the transmission of the `Delay_Req` message has a similar effect as the delay in the `Sync` message, for the reception time used in (5) would be increased, leading to a greater one-way delay in (2). Yet, delay attack in the `Delay_Req` message can cause a more significant disruption of the synchronization process, because the calculation of the

one-way delay is not done as frequently as the offset correction synchronization process. An incorrect value of one-way delay can cause errors in all upcoming offset calculations in (1).

Finally, if the `Delay_Response` message is not received back at the slave clock after a fixed delay request interval, the whole calibration process of the one-way delay would be voided. An adversary can even launch this attack, and then add in more delay fluctuations, such as new network components, to the system. Since the one-way delay is not being recalculated due to the timeout of the delay request interval, the additional delay from the new components is not being accounted for in the calculation of (1).

Suggested countermeasures: PTP can be modified to have a backup plan to compensate for the missing or delayed messages. For example, by taking the averages of the `Delay_Req` and `Delay_Response` messages in combination with previous values, the effects of timed out or postponed messages can be reduced. Also, there should be an algorithm in PTP to determine any abnormal values of the timestamps in the messages. If any abnormality is found for an extended period of time (e.g., the last 4 out of 8 of the past one-way delays are significantly higher than the first 4 values), validation of the data against other neighboring nodes might be helpful.

3.4 Replay

Goal: Any of the following: (a) create congestion in the network stacks of the clocks (b) desynchronize clocks

Attack: Record legitimate message(s) being transmitted on the communication path and, at a later time, slightly modify and then re-inject the recorded message(s) into the network.

Effects: The replayed message(s) would have to contain a greater `sequenceID` than the previous message, as this value would be checked against the
`parent_last_sync_sequence_number` stored in the data sets of the receiving node. This is why slight modification of the replayed messages is required. Since PTP does not specify integrity protection, `sequenceID`s can easily be faked.

These replayed messages would be interpreted as genuine messages. Any state or data change of occurrence of events caused by the messages would be processed in a First-In-First-Out order. For example, when the `Sync` message is being replayed to the slave clock, the slave clock would record the precise reception time of it. However, we were uncertain how the precise reception time is stored. If there is one storage location for it, then the later replayed message would overwrite the precise reception time of the first `Sync` message. If there are multiple storage places, then the precise recorded times can be queued up and would not lead to a problem.

Furthermore, replaying messages can saturate the processing queue at the clocks and congest their network stacks, which may result in dropping authentic synchronization messages from the true master clocks.

Suggested countermeasures: Use an integrity protected network path, e.g., a VPN connection, to prevent message being spoofed or injected into the network. Alternatively, use a capable message authentication mechanism to ensure the data origin integrity of the PTP messages.

3.5 Denial of Service (DoS)

Goal: To deny PTP clock(s) time synchronization service.

Attack: In addition to generic DoS attacks through flooding networks and overflowing communication stacks, we identified a protocol-specific way of denying service to PTP slave clocks. The significance of this attack is that generic IDS or other techniques for detecting DoS are unlikely to detect it.

The purpose of this attack is to trick the attacked slave clock into rejecting `Sync` and `Follow_Up` messages from the true master clock. Since the attack is the same for both types of messages, we explain it using the `Sync` message case.

For performing DoS using `Sync` messages, the adversary first spoofs the victim slave clock with a `Sync` message using the true master clock address, albeit with a `sequenceID` value greater than the one in the previous `Sync` message. Upon processing this message, the victim clock updates the `parent_last_sync_sequence` in its parent data set accordingly. Denial of service occurs when the true master sends its next `Sync` message to the victim. Since the `parent_last_sync_sequence` in the slave clock has already been incremented upon receiving the spoofed `Sync` message, the `Sync` message from the true master clock has `sequenceID` value less than or at most equal to the `parent_last_sync_sequence`. Therefore, this `Sync` message is rejected by the victim slave without the master being notified. No synchronization can take place until the true masters `Sync` message has a `sequenceID` once again greater than the `parent_last_sync_sequence` in the slave clock. The greater the `sequenceID` in the adversarys `Sync` messages, the longer the time synchronization service is denied. If the adversary keeps on sending `Sync` messages to increase the victims value of `parent_last_sync_sequence` before the true master can catch up with it, the victim can be permanently deprived of synchronization with the true master. The adversary starts controlling the time of the slave clock without the victim or its master clock realizing this.

Effects: A small-scale denial of service attack, which means only a few nodes of the network are affected, may cause the synchronization to be less accurate across the system. If only slave clocks are affected, these slave clocks may then be set to run on a local clock, drifting away from each other due to the differences in each clocks skew. The bigger the scale of the attack, the more PTP clocks would have to run on local clocks. Furthermore, the tree-like hierarchical organization of PTP networks allows attacker(s) to increase the scale of the attack dramatically by denying synchronization to just few boundary clocks that are close to the GMC.

Suggested countermeasures: Employ message authenticity and integrity protection or use port-level security to limit the set of the devices authorized to send synchronization mes-

sages. The limitations and drawbacks of both types of countermeasures have been discussed in the previous sections.

4 Summary of Threats and Countermeasures

Results of our analysis suggest that the PTP protocol is weak in ensuring the integrity and authenticity of the messages being transmitted. Adversaries can easily use this weakness to perform generic attacks, such as flooding the victims, removing or modifying PTP messages.

More importantly, we showed that an adversary can mount PTP-specific attacks, which generic intrusion detection systems are unlikely to spot without performing deep inspection and maintaining the protocol state. Because each PTP clock elects its master autonomously using the content of recent synchronization messages, an insider can easily win the best master clock algorithm through observing the PTP traffic and then injecting a winning Sync message. After becoming a rogue master, an attacker can control the victim(s) clock(s). Furthermore, being able to modify just one field in PTP synchronization messages, an attacker can cause PTP-specific denial of service to the victim clock in a way noticeable neither by the victim slave nor its master. An attacker can also disrupt a PTP hierarchy by spoofing synchronization messages. Additionally, the current protocol lacks a mechanism for detecting and compensating for out of range data that can result in an inconsistent state of a PTP device or even the whole network.

The most straightforward countermeasure against the bulk of the above threats is the use of message integrity protection mechanisms based on cryptographic techniques. Unfortunately, cryptographic techniques might introduce prohibitive delays or, even worse, increase the delay fluctuations, which could be fatal to the protocol participants, for whom sub-microsecond precision is paramount. For example, the time to encrypt using public key cryptography depends on properties of the key (such as number of 1 bits) [21]. With classical cryptosystems, though, the time to encrypt is constant but the key management problems are worse. Key management in PTP networks could be very hard even with public keys, as the protocol does not have any provisions for registering slave or master clocks, or binding their identities to keys. Further investigation is needed to decide which cryptographic integrity protection best suits PTP.

An alternative countermeasure is to employ port-level security [20] to restrict the set of devices that can perform the role of master clocks. Albeit an incomplete solution, these access controls might be more effective performance-wise than those based on cryptography. In addition, key management can be avoided. However, such controls reduce the robustness of the PTP in the presence of network failures. Since PTP clocks locally elect best master clocks for time synchronization, the role of master clock can be passed from clock to clock as the network topology changes due to device additions and failures. Besides making a PTP network more fragile, port-level security also increases the overhead of configuring network switches, and, as a result, increases the chances of friendly denial of service due to configuration errors. Last but not least, physical isolation could be an option. One can argue that access by adversaries to critical PTP networks can be practically eliminated through physical security. However, physical isolation is not always possible in the domains where PTP is necessary, thus maintaining the insider threat at a non-negligible level.

5 Conclusions and Future Work

IEEE 1588, also known as PTP, is a valuable protocol for synchronizing clocks in a local area network. Being able to provide synchronization accuracy in the sub-microsecond range, the protocol is being deployed in a wide range of application domains, from factory automation to avionics and military systems. However, the protocol lacks efficient security mechanisms to ensure the integrity of transmitted messages and to validate the trustworthiness of the sender. We analyzed the effects of five different types of attacks on a PTP network: modification, masquerading, delay, replay and, denial of service. It can be seen that PTP alone is weak against these attacks, and that additional security mechanisms are necessary to protect a PTP network. Damage caused by failure in time synchronization of some critical industrial systems is unforeseeable.

We did not implement the discovered attacks or countermeasures due to the lack of publicly available PTP implementations. When the situation changes, developing a proof of concept for our attacks and countermeasures could be very helpful for evaluating their feasibility. Because of the critical nature of the PTP application domains [18, 19], we prefer to make the community of security professionals aware of our results now.

We limited our analysis to the synchronization messages. The damage caused by corrupting the management messages or using them to launch attacks (e.g., to change information on data sets of PTP clocks) can be much greater than the time synchronization messages, because these messages have the capability to control all of the data sets and the status of a PTP clock. Analysis of management messages is a promising avenue for future research. Another interesting direction is to validate the feasibility of cryptographic countermeasures as applied to PTP.

Acknowledgments.

Authors would like to thank the following people who provided their comments on the earlier revisions of the paper: Galina Antonova, Matt Bishop, and John Eidson. Craig Wilson helped us to improve the readability of the paper.

References

[1] A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Standard 1588-2002, November 2002.

[2] National Institute of Standards and Technology, IEEE 1588 Website, `http://ieee1588.nist.gov/`

[3] Matt Bishop, Introduction to Computer Security. Toronto, ON: Addison-Wesley, 2005, pp. 7.

[4] M. Bishop, "A Security Analysis of the NTP Protocol Version 2," Sixth Annual Computer Security Conference Proceedings pp. 20-29 (Dec. 1990).

[5] P. Doyle, "Introduction to Real-Time Ethernet I, The Extension A Technical Supplement to Control Network, vol.5, issue. 3, May-June 2004. Available: www.ccontrols.com/pdf/Extv5n3.pdf

[6] P. Doyle, Introduction to Real-Time Ethernet II, The Extension A Technical Supplement to Control Network, vol.5, issue. 4, July-August 2004. Available: www.ccontrols.com/pdf/Extv5n4.pdf

[7] J. Eidson, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," presented at IEEE SCV Instrumentation and Measurement Society meeting July 16 2003, Sunnyvale, CA.

[8] J. Eidson, M.C. Fischer, and J. White, "IEEE-1588 Standard for a precision clock synchronization protocol for networked measurement and control systems, " Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting, December 3-5, 2002, Reston, Virginia.

[9] A. James, "Using IEEE 802.1x to Enhance Network Security," The Foundry Network, October 2002. Available: http://www.foundrynet.com/solutions/appNotes/PDFs/802.1xWhite_Paper.pdf

[10] A. Mishra and W. A. Arbaugh, "An Initial Security Analysis of the IEEE 802.1x Standard," Tech. Rep. CS-TR-4328, University of Maryland, College Park, February 2002. Available: http://www.cs.umd.edu/~waa/pubs/1x.pdf

[11] D. Mohl, "IEEE 1588-Precise Time Synchronization as the Basis for Real Time Application in Automation," Available: www.industrialnetworking.com/support/general_faqs_info/Precise_Time_Sync.pdf

[12] A. S. Tanenbaum, Computer Networks. Upper Saddle River, New Jersey: Prentice Hall PTR, 2003.

[13] "Proceedings of the 2004 IEEE-1588 Conference, September 27-9, 2004," September 2004, Gaithersburg, MD, pp. 9 -10. Available: http://ieee1588.nist.gov/Conference_2004_proceedings.pdf

[14] IEC, "Precision clock synchronization protocol for networked measurement and control systems" IEC 61588, First Edition, 2004, pp. 158.

[15] "NTP: Network Time Protocol," Available: www.ntp.org

[16] D. Deeths and G. Bruneet, "Using NTP to Control and Synchronize System Clocks: Part 1: Introduction to NTP, " Sun BluePrints Online, July 2001. Available: www.sun.com/blueprints/0701/NTP.pdf

[17] N. Dunbar, "IPsec Networking Standards An Overview, " Information Security Technical Report, Volume 5, Issue 1, 1 March 2001, pp.35-48.

[18] K. R. Harris, S. Balasubramanian, and A. Moldovansky, "The Application of IEEE 1588 to a Distributed Motion Control System," presented at the ODVA CIP Networks Conference, Nov. 16-18, 2004. Available: http://ieee1588.nist.gov/z_RA_2_Motion_Control_Application_Using_1588.pdf

[19] G. Gaderere, T. Sauter, G. Bumiller, "Clock Synchronization in Powerline Networks," Proceedings of International Symposium on Power Line Communications and Its Application, 2005, pp. 71-75.

[20] J. B. Hansen and S. Young, The Hacker's Handbook, CRC Press, 2004, pp.512.

[21] M. Bishop. Private communication.

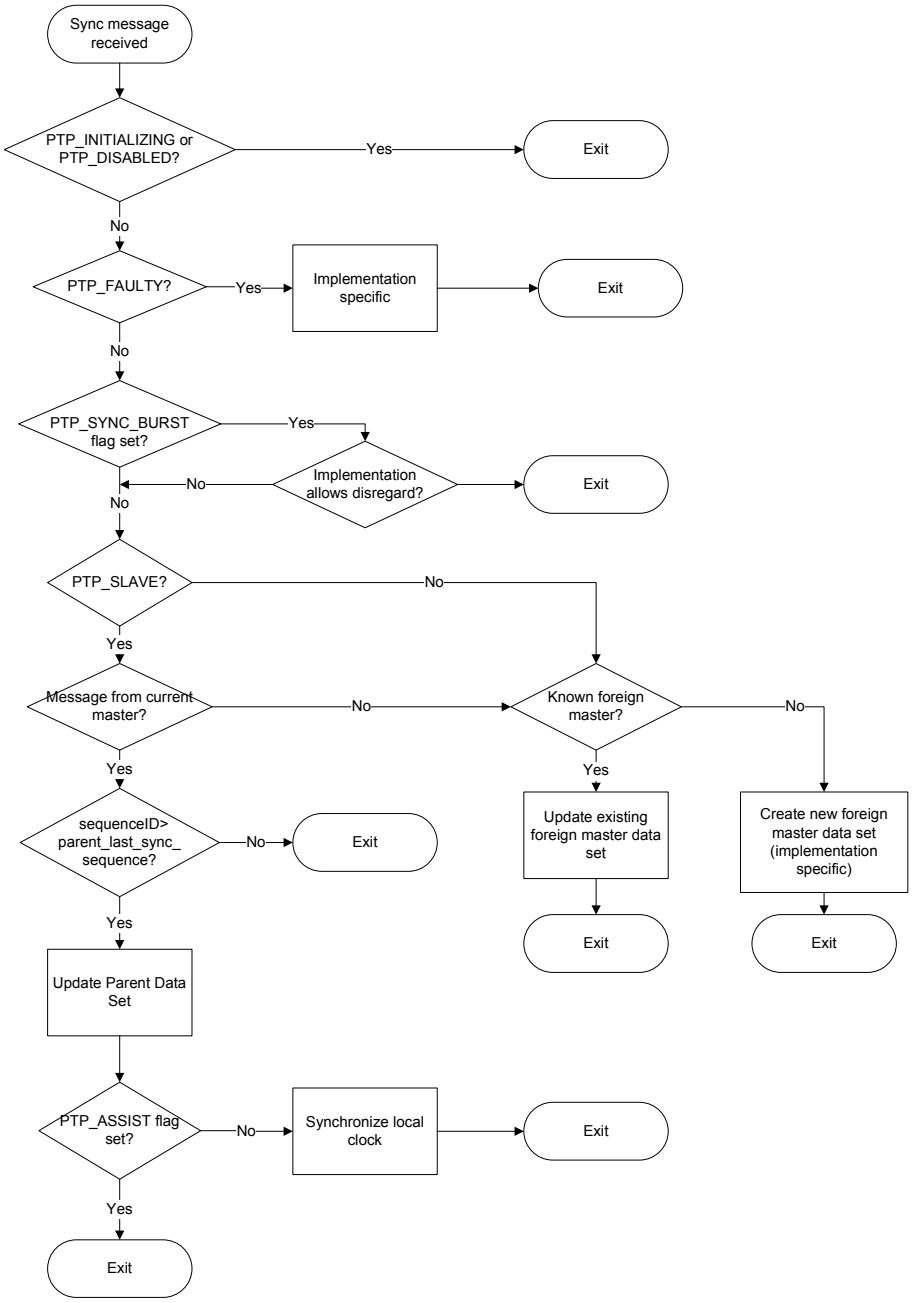
[22] K. Harris, S. Balasubramanian and A. Moldovansky, "Cars and Robots Are One - Develop Motion Control Solutions Over Standard Networks Such As Ethernet," InTech Volume 52, Issue 22, 1 February 2005, Instrument Society of America, pp.28-31.

[23] M. Manzo, T. Roosta and S. Sastry, "Time Synchronization Attacks in Sensor Networks," in Proceedings of the 3rd ACM Workshop on Security of ad hoc and Sensor Networks, Alexandria, VA, November 2005.

[24] K. Sun, P. Ning, C. Wang, A. Liu and Y. Zhou, "TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks," in Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06), Alexandria, VA, November 2006.

[25] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. D. Tygar, "SPINS: Security Protocols for Sensor Netowrks," Mobile Computing and Networking, Rome, Italy, 2001.

Appendix A Processing of `Sync` message by PTP clock in slave mode.



It is assumed that the reception node is in an active state and that it is not faulty.

Appendix B Field names of `Sync` and `Delay_Req` messages

*VersionPTP | versionNetwork	grandmasterClockVariance
*Subdomain	grandmasterClockPreferred | grandmasterIsBoundaryClock
*messageType | sourceCommunicationTechnology | sourceUuid	syncInterval
*sequenceId	localClockVariance
*control | *reserved | *flags	localStepsRemoved
delayReceiptTimestamps (seconds)	localClockStratum
originTimestamp	localClockIdentifier
epochNumber | currentUTCOffset	parentCommunicationTechnology | parentUuid
grandmasterClockUuid	parentPortField
grandmasterPortId | grandmasterSequenceId	estimatedMasterVariance
grandmasterClockStratum	estimatedMasterDrift
grandmasterClockIdentifier	utcReasonable

* indicates field that is also in the `Follow_Up` and `Delay_Response` messages

Additional fields in a `Follow_Up message`: associatedSequenceId, preciseOriginTimestamp

Additional fields in a `Delay_Response` message: delayReceiptTimestamp, requestingSourceCommunicationTechnology, requestingSourceUuid, requestingSourcePortId, requestingSourceSequenceId