MS-PTP: Protecting Network Timing from Byzantine Attacks

Shanghao Shi[1], Yang Xiao[2], Changlai Du[1], Md Hasan Shahriar[1], Ao Li[3], Ning Zhang[3], Y. Thomas Hou[1], and Wenjing Lou[1]

Virginia Tech[1]

University of Kentucky[2]

Washington University in St. Louis[3]

VIRGINIA TECH™

Why Time Synchronization is Important?

- Many time-sensitive applications impose stringent time synchronization requirements. Failure to meet these requirements causes significant consequences.

The 5G URLLC requires a 1.5 us level synchronization to avoid performance degradation and communication failures [1].

Milliseconds-level de-synchronization among servers in high-performance computing clusters trigger a 100x performance degradation [2].

IEEE time-sensitive networking standard (802.1) requires its clocks to be synchronized within 1 us [3].

Sensor de-synchronization in autonomous driving cars can cause temporal displacements in critical control states and cause safety hazards to humans [4].

[1] 3GPP, “Study on the enhancement of ultra-reliable low-latency communication (urllc) support in the 5g core network (5gc),” Tech. Rep., 6 2019,3GPP TR23.725 V16.2.0 (Release16).
[2] O. Obleukhov and A. Byagowi, “How precision time protocol is being deployed at meta,” 2022. [Online]. vailable: https://engineering.fb.com/2022/11/21/productionengineering/precision-time-protocol-at-meta/
[3] “Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications,” IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011), pp. 1–421, 2020.
[4] A. Li, J. Wang, and N. Zhang, “Chronos: Timing interference as a new attack vector on autonomous cyber-physical systems,” in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2426–2428.

Time Synchronization Mechanisms

- Currently there are three well-received time synchronization mechanisms: the network time protocol (NTP), the precision time protocol (PTP), and GPS-based mechanisms.

	GPS	NTP	PTP [5]
Accuracy	~100ns	~50ms	<100us
Service area	Outdoor only, need GPS antenna	Internet	Local-area networks
Work mode	Broadcast	Client-Server	Client-Server
Use case	Mobile phones	Desktop/ Laptop	Time- and latency-sensitive networks

Table 1: The comparison of different time synchronization mechanisms.

- In this work, we focus on the **precision time protocol** (PTP), because it is the de-facto time synchronization mechanism used by various time- and latency-sensitive networks.

[5] J. C. Eidson, M. Fischer, and J. White, “Ieee-1588™ standard for a precision clock synchronization protocol for networked measurement and control systems,” in Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting, 2002, pp. 243–254.

Precision Time Protocol Overview

- The Precision Time Protocol (PTP), originally developed by the IEEE 1588 working group [4], is widely regarded as the de-facto solution to highly accurate clock synchronization.
  - PTP guarantees a sub-microsecond level accuracy.
  - PTP has higher accuracy than network time protocol (NTP) and better flexibility than GPS.
  - PTP establishes a tree-structured, master-slave hierarchy.
  - The reference timing information is transferred from PTP server to its clients through a two-way time transfer mechanism.
  - Many industrial standards have specified PTP as their time synchronization mechanism.

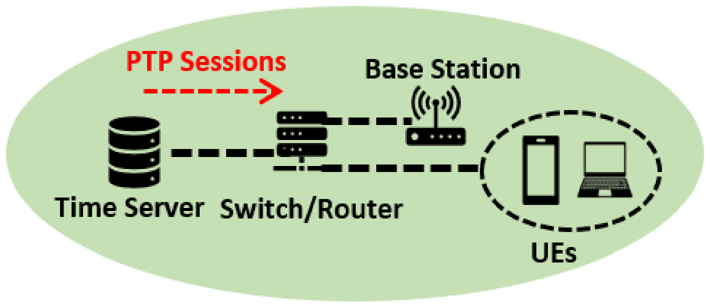


Fig 2: The client-server work mode of PTP.

Industrial Fields	Standards
Time Sensitive Networking (TSN)	[3]
Smart grids	[6]
Telecom	ITUT 8265.1
Automotive	[3]

Table 2: Industrial standards use PTP to provide timing service.

[6] "Ieee standard profile for use of ieee 1588 precision time protocol in power system applications," IEEE Std C37.238-2017 (Revision of IEEE Std C37.238-2011), pp. 1–42, 2017.

Elect the Best Time Master

- PTP establishes a tree-structured, master-slave hierarchy.
  - Within PTP's terminology, the most accurate clock--the grandmaster clock (GM), is elected as the unique time server; The intermediate routers and servers are known as boundary clocks (BCs) or transparent clocks (TCs); And the end leaves are ordinary clocks(OCs).
  - PTP specifies a dynamic leader election procedure to build up this architecture.
    - Every node proactively listens for a specific message—the ANNOUNCE message that contains the identity and clock quality of its sender.
    - Upon reception, the node determines the master-slave relationship with the message sender by a pre-defined best master clock algorithm (BMCA).

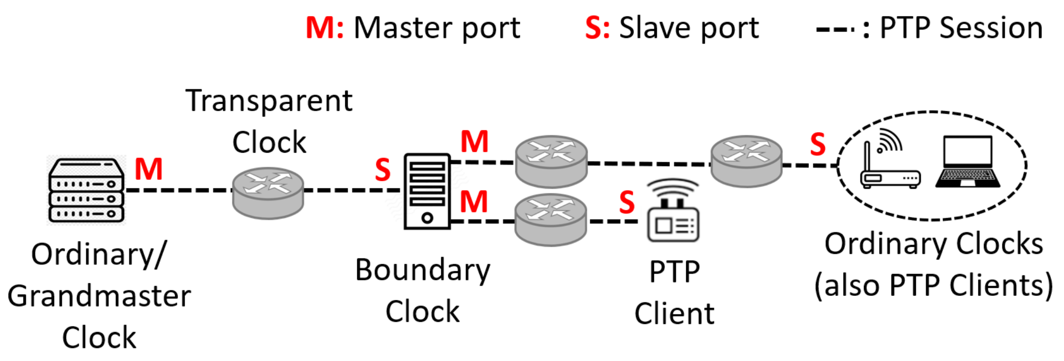
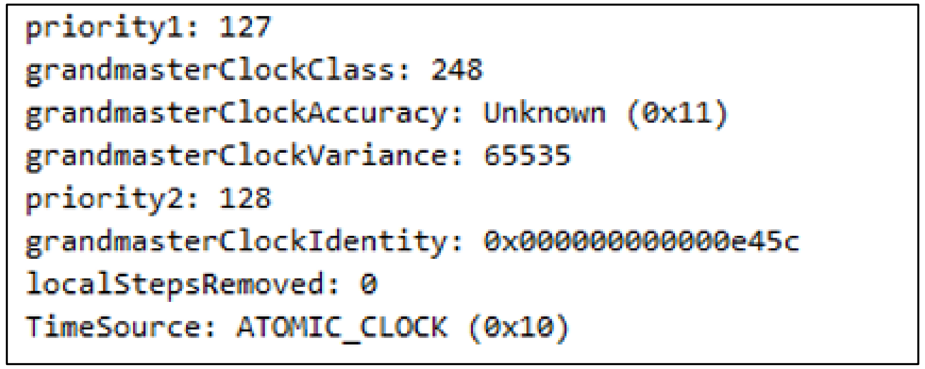


Fig 3: PTP network hierarchy.



```
priority1: 127
grandmasterClockClass: 248
grandmasterClockAccuracy: Unknown (0x11)
grandmasterClockVariance: 65535
priority2: 128
grandmasterClockIdentity: 0x000000000000e45c
localStepsRemoved: 0
TimeSource: ATOMIC_CLOCK (0x10)
```

Fig 4: The clock information contained in the ANNOUNCE message.

Two Way Time Transfer

- PTP utilizes a two-way time transfer method to transfer timing information.

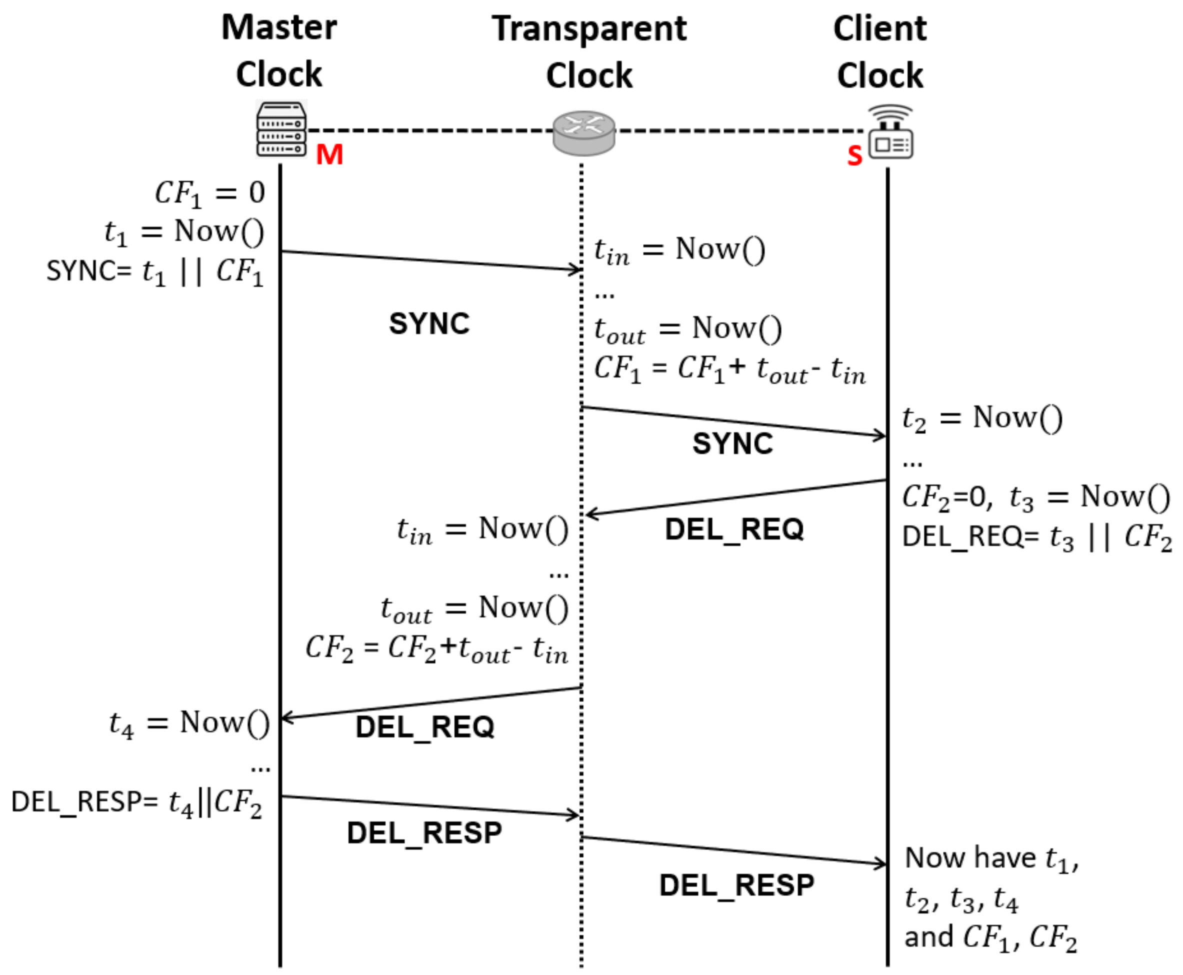


Fig 5: TWTT workflow.

- The client measures its offset and skew with the master clock and calibrates its clock.
- Standard RTT estimator:
  - $\text{offset} = ((t_4 - CF_2 - t_3) - (t_2 - CF_1 - t_1))/2$
  - $\text{skew} = (\text{offset}_n - \text{offset}_1)/(t_n - t_1)$
- Except from the standard RTT estimator, there are other methods such as least-square estimator and Kalman filter.

Existing Work

- The vanilla version of PTP was designed decades ago and has no built-in security mechanism. As a result, PTP can be easily disrupted by simple network-level time-shifting attacks through message spoofing and modification [7, 8, 9].
- Luckily, currently there are several works focusing on addressing them:
  - The latest version of PTP recommends group key-based direct authentication and TELSA-based delayed authentication to support message authentication [10].
  - [9] proposes an elliptic curve-based and public-key signature scheme to establish the authenticity of network clocks.
  - [11] specifies a key management scheme to help establish message authentication.
- But little attention has been paid to insider adversaries, who are compromised legitimate participants of the system. They act as **Byzantine insiders** and the current mechanisms fail to address them.

[7] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert, “Impact of cyberattacks on precision time protocol,” IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 5, pp. 2172–2181, 2019.
[8] W. Alghamdi and M. Schukat, “Cyber attacks on precision time protocol networks—a case study,” Electronics, vol. 9, no. 9, p. 1398, 2020.
[9] E. Itkin and A. Wool, “A security analysis and revised security extension for the precision time protocol,” IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 1, pp. 22–34, 2017.
[10] Ieee standard for a precision clock synchronization protocol for networked measurement and control systems,” IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008), pp. 1–499, 2020.
[11] NTS4PTP - Key Management System for the Precision Time Protocol Based on the Network Time Security Protocol. Available at: https://www.ietf.org/id/draft-langer-ntp-nts-for-ptp-04.html

What about Insider Attack?

- In this work, we first focus on investigating insider attacks. We are going to answer the following 4 questions:
  - (a) Can an insider node jeopardize the operation of others in the current PTP networks?
  - (b) If so, to what extent can they infect the time of the victim node?
  - (c) Can the current defense mechanisms counter these attacks?
  - (d) What is the consequence caused by insider attacks on real systems?

Attack Settings: Testbed and Attack Model

- We deploy a real Raspberry Pi 4 testbed to demonstrate and evaluate our attack.
- We do not assume the attacker to be the existing grandmaster or Man-in-the-middle attacker, for this makes the attack trivial. We also assume an **authentication mechanism is in place** and the attack shall bypass it by design.
- We assume the attacker controls one client node and it is able to know essential operation parameters and secret keys, as well as monitor PTP traffic and send malicious packets.

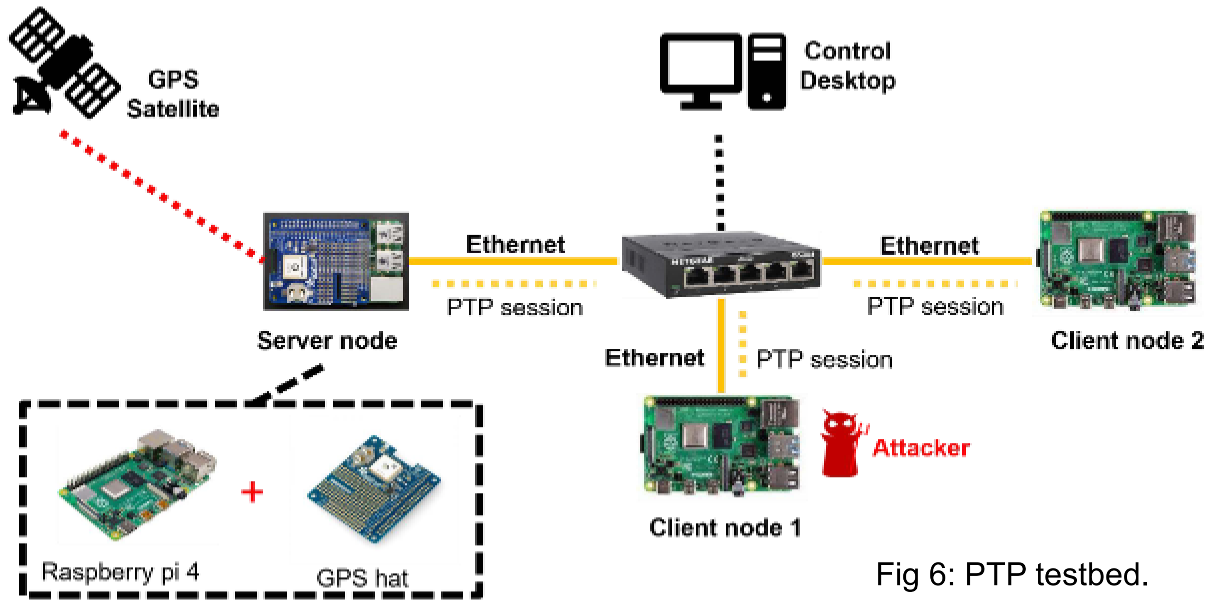


Fig 6: PTP testbed.

Attack settings: Target and Strategies

- The victim implementations are current well-known PTP software:
  - PTPD Daemon [12]
  - Linuxptp [13]: Default profile, telecom profile, and unicast profile.
- The attacker's goal is to shift the system time of a chosen victim node and we investigate the attacker's capability by letting the attacker add the malicious delay d constantly (2 seconds), randomly (mean 1.5 seconds, standard deviation (std) 0.5 second) and cumulatively (25 milliseconds per packet) over a 120 seconds attack period.

[12] PTPD Daemon, https://www.ibm.com/docs/zh/aix/7.1?topic=p-ptpd-daemon.
[13] Linux, "An implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux." 2011. [Online]. Available: https://linuxptp.sourceforge.net/

Attack Approach (1)

- We carry out a time-shifting attack in 4 steps:
  - **Phase 1: Clock information extraction**: The attacker n_{adv} reads the current transmitting ANNOUNCE message ANN_h on its UDP port 320 and extracts the current GM's clock quality information q_h from the message payload.
  - **Phase 2: Priority inversion**: The attacker adjusts the clock quality information according to the BMCA-defined clock quality comparison rules, such as increasing the clock priority level by one to generate a better clock quality information as q_{adv}. The attacker generates a malicious ANNOUNCE message ANN_{adv} according to the authentication mechanism used in the system. For the digital signature-based method, $ANN_{adv} = pk_{n_i}||q_{adv}\,||sig_{sk_{n_i}}$. For the symmetric key-based method, $ANN_{adv} = q_{adv}||MAC_{\mathrm{sk}_{n_i}}$.

Attack Approach(2)

- **Phase 3: Injection and confirmation:** The attacker broadcasts ANN_{adv} periodically through UDP port 320. At the same time, the attacker monitors the incoming messages on this port and if the attacker fails to receive any incoming ANNOUNCE message for a certain time interval T_{adv}, there is a high probability that nodes in the network have already taken n_{adv} as the new grandmaster.
- **Phase 4: Time shifting:** After the confirmation, the attacker starts the PTP engine to send erroneous information. It follows the normal PTP workflow (i.e. TWTT) but modifies the timestamp field of the SYNC message only to its victim by adding a delay d to t_1. d can be a constant delay, a random delay, or a cumulative increasing delay. By protocol, the offset measured at the victim device is shifted by $d/2$. This malicious $SYNC_{adv}$ message is also attached with a proper digital signature or message authentication code generated by the shared credentials between the attacker and the victim node.

Attack Results

- The attacker successfully shifted the system time of victim nodes in different settings!

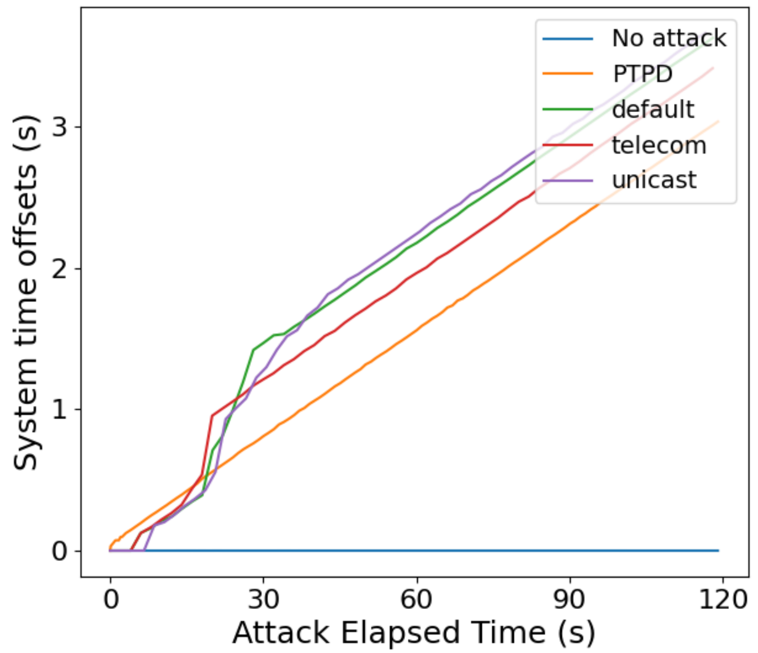


Fig 7: Adding Cumulative Delays.

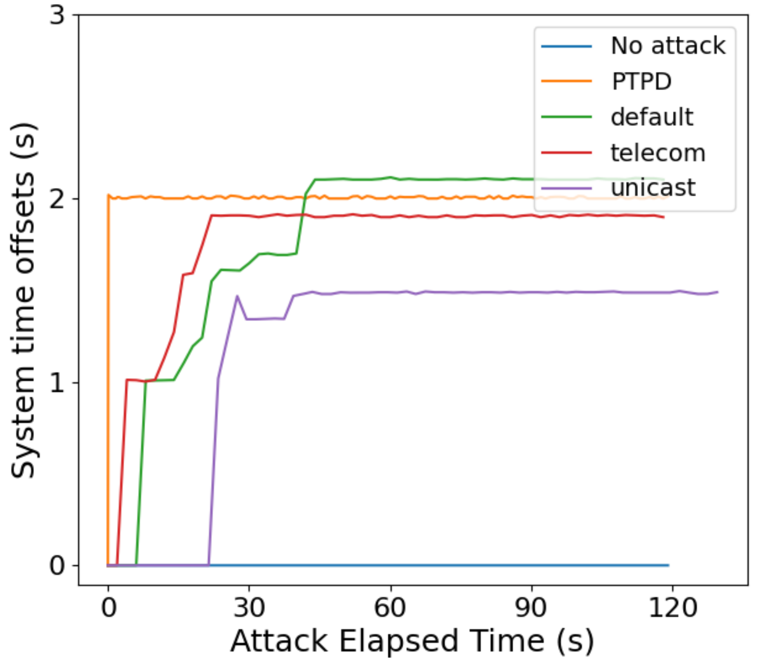


Fig 8: Adding Constant Delays.

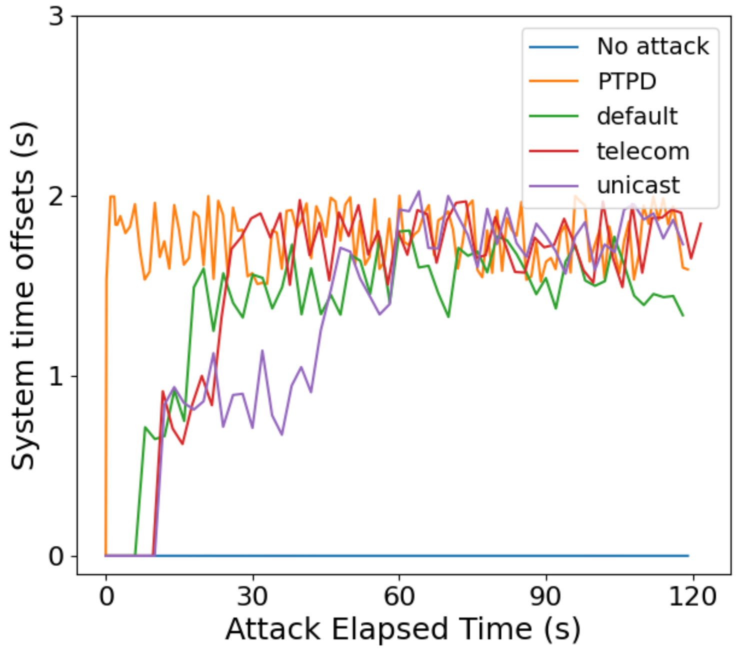
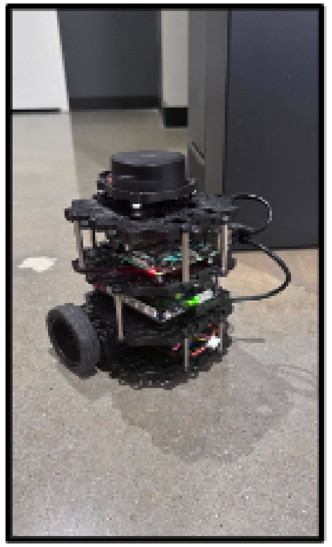
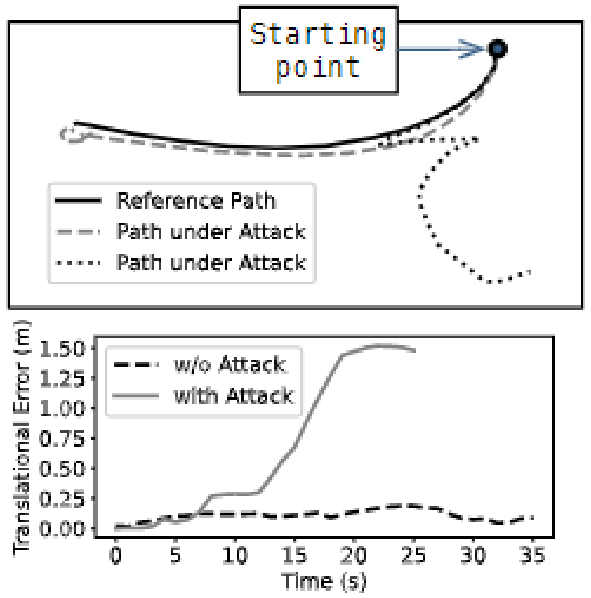


Fig 9: Adding Random Delays.

Case study: Timing Attack on a Real System

- We investigate the effect of timing attack on a real Turtlebot 3 robotic pltform, who receives sensor inputs and commands from the cloud.
- Our timing attack de-synchronize the sensor inputs and commands to the robot.
- In our experiment, the robot fails to localize and control itself and goes to a false trajectory.


(a) Turtlebot3.



(b) Control deviations on trajectories.

Fig 10: PTP attack on robotic platforms

Lesson learned

- The attack has shown that a PTP network relying on a single-time source can be easily disrupted by any **Byzantine insider** existing on the server-client communication path.
- We adopt the fundamental idea of using **multiple servers** to counter Byzantine sources from the very famous Byzantine fault tolerance state machine replication (BFT-SMR) scheme.
- This idea is in line with PTP's developing trend--the newest version has already dictated the use of redundant servers to build robust synchronization services.
- Fortunately, current PTP leaves room for the simultaneous existence of multiple servers. This can be achieved by properly configuring the PTP software’s $.conf$ file. Each PTP client can open up multiple PTP sessions, with each one having its own *domain number*. By design different PTP sessions in different domains do not cause interference to others and within each domain, there can be a time server.

MS-PTP System Model

- Network Model:
  - We assume there are m different servers simultaneously exist in the network and an individual client can fetch n of them. Among these n PTP sessions, f of them are compromised by insider adversaries.
  - As a result, f of the measurements received by a client is byzantine, denoted by $(d_1, d_2, \ldots, d_{n-f}, b_1, \ldots, b_f)$. The honest measurements $d_1, d_2, \ldots, d_{n-f}$ follows gaussian distribution $d_i \sim N\left(g, \sigma_i^2\right)$. The general representation of a measurement (without knowing its honesty or not) is v_i $(i = 1,2, \ldots, n)$.
  - The system's time synchronization requirement for a client is represented as r.

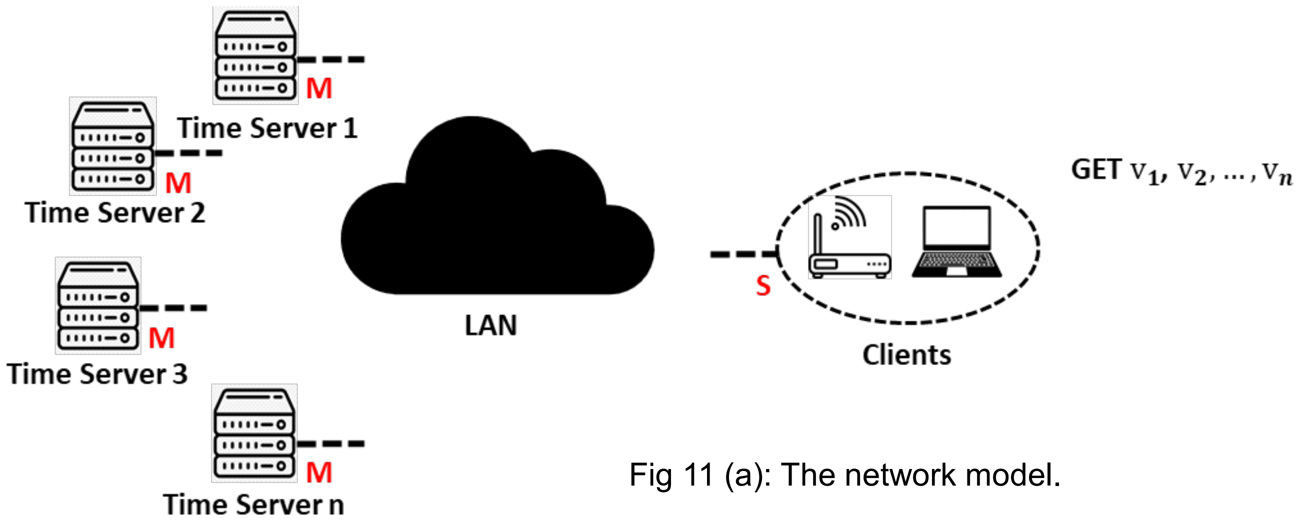


Fig 11 (a): The network model.

MS-PTP Threat Model

- Threat Model:
  - An insider attacker can exhibit arbitrary behaviors such as delaying, intercepting, manipulating and replaying the packets. Insider attackers can also collude with each other. Consequently, any time synchronization session involving an insider attacker becomes a Byzantine session that delivers arbitrary timing measurement to the client (i.e. the byzantine measurements $b_1, \dots, b_f$ follows arbitrary distribution).
  - The only limitation for the adversary‘s capability is that they are the minority of the total population and the number of them is smaller than a threshold f.

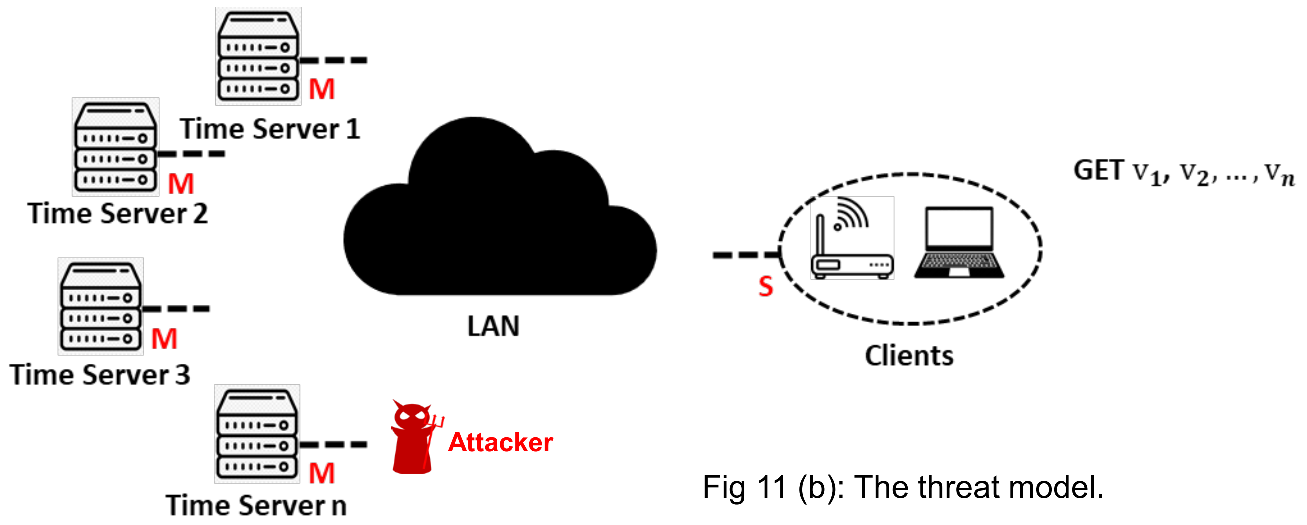


Fig 11 (b): The threat model.

MS-PTP System Assumption

- Uncertainty level assumption:
  - The standard deviation σ_i of an honest session is considered significantly smaller than the system's synchronization requirement r.
  - The probability that a measurement from an honest session violates the requirement r is $\mathrm{P}(|d_i - g| > r) = 1 - \frac{1}{\sqrt{2\pi}\sigma_i}\int_{-\mathrm{r}}^{r} e^{-\frac{x^2}{2\sigma_i^2}} dx$ or in the form of gaussian error function $\mathrm{P}(|d_i - g| > r)$=erfc($\frac{r}{\sqrt{2}\sigma_i}$).
  - This probability is considered as the unreliability rate of the system and is considerable small. For example, to achieve 5G URLLC's 10^{-7}error rate, r satisfies $r > 5.3\ \sigma_i$.

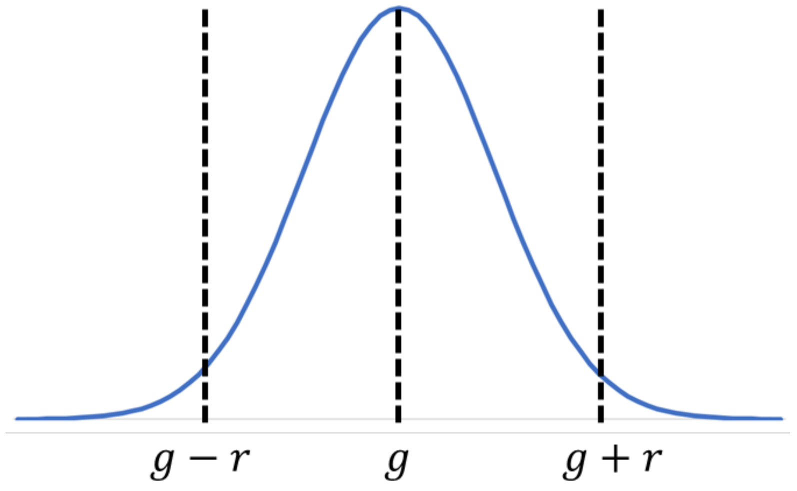


Fig 12: The probability distribution of honest sessions.

MS-PTP System Assumption

- System Goal:
  - The goal is to establish a fault-tolerant time synchronization mechanism against insider adversaries when more than 2/3-majority of sessions are honest ($n \geq 3f + 1$).
- Definition of robustness:
  - An aggregation rule $\mathrm{A}(m_1, m_2, \ldots, m_{n-f}, b_1, \ldots, b_f)$ is r-robust when its output o satisfies $\|E[o] - g\| \leq s < r$, where s is a non-parametric bound and r is the system's requirement.

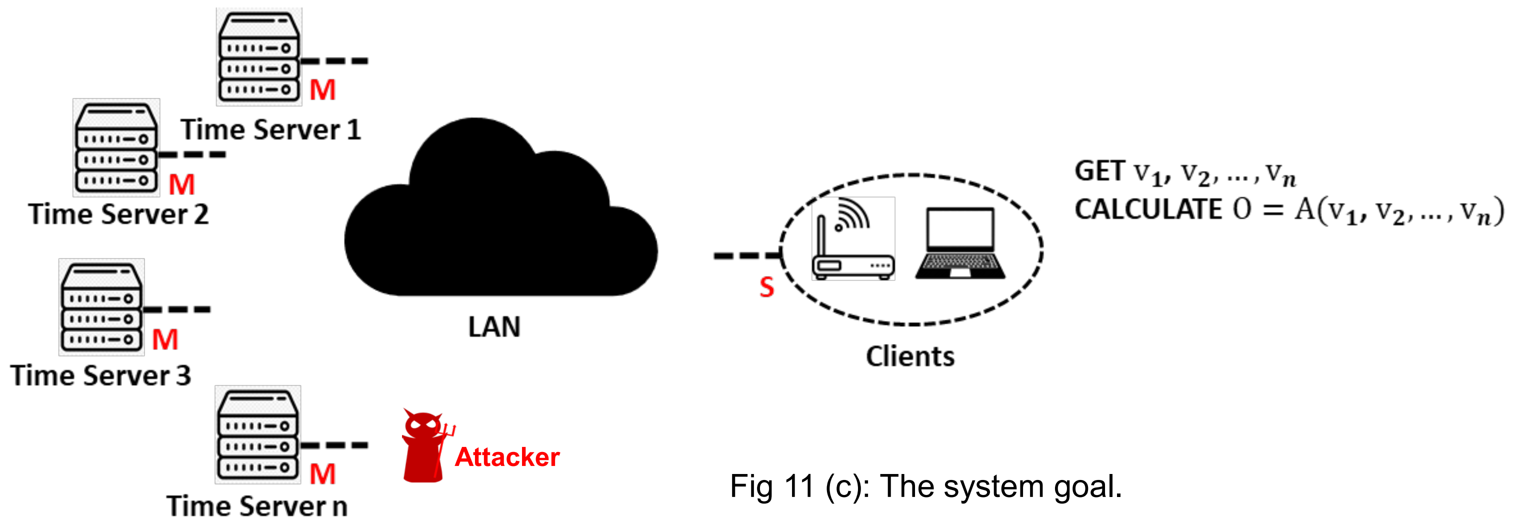
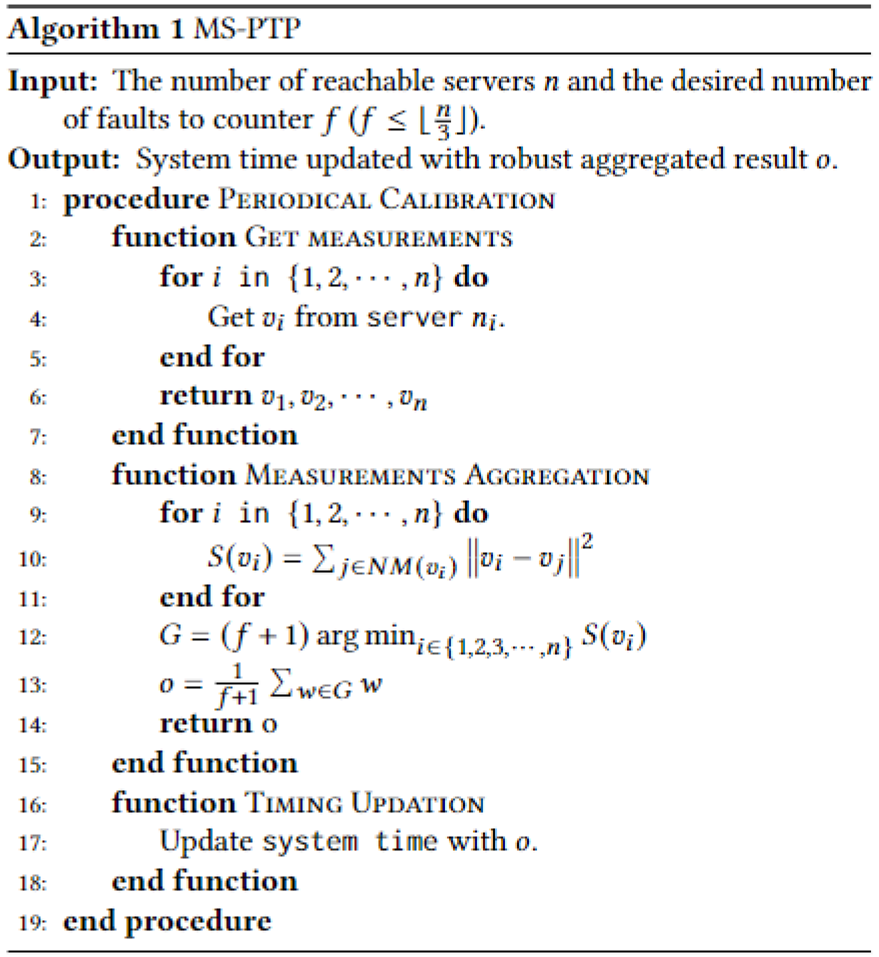

Time Server 1
M
Time Server 2
M
Time Server 3
M
Time Server n
M
LAN
Attacker
S
Clients
GET $\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_n$
CALCULATE $\mathrm{O} = \mathrm{A}(\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_n)$

Fig 11 (c): The system goal.

MS-PTP Aggregation Rule

- **The aggregation rule**: We define a score $S(v_i) = \sum_{j \in NM(v_i)} ||v_i - v_j||^2$, where $NM(v_i)$ includes the $2f$ nearest measurements of v_i. The aggregation rule is:
- $$G = (\mathrm{f}+1)argmin_{(i=1,2,\ldots,n)} S(v_i)$$
- $$o = \frac{1}{f+1}\sum_{w \in G} w$$
- **Performance guarantee:** We mathematically prove that the accuracy of the aggregation output o is bounded by $||E[o] - g|| \leq \sqrt{2}\sigma_{max}$, comparable to the output of honest sessions without attacks.



Algorithm 1 MS-PTP

Input: The number of reachable servers n and the desired number of faults to counter f ($f \leq \lfloor \frac{n}{3} \rfloor$).
Output: System time updated with robust aggregated result o.

```
procedure PERIODICAL CALIBRATION
    function GET MEASUREMENTS
        for $i$ in $\{1, 2, \cdots, n\}$ do
            Get $v_i$ from server $n_i$.
        end for
        return $v_1, v_2, \cdots, v_n$
    end function
    function MEASUREMENTS AGGREGATION
        for $i$ in $\{1, 2, \cdots, n\}$ do
            $S(v_i) = \sum_{j \in NM(v_i)} \|v_i - v_j\|^2$
        end for
        $G = (f+1) \arg\min_{i \in \{1,2,3,\cdots,n\}} S(v_i)$
        $o = \frac{1}{f+1} \sum_{w \in G} w$
        return o
    end function
    function TIMING UPDATION
        Update system time with $o$.
    end function
end procedure
```

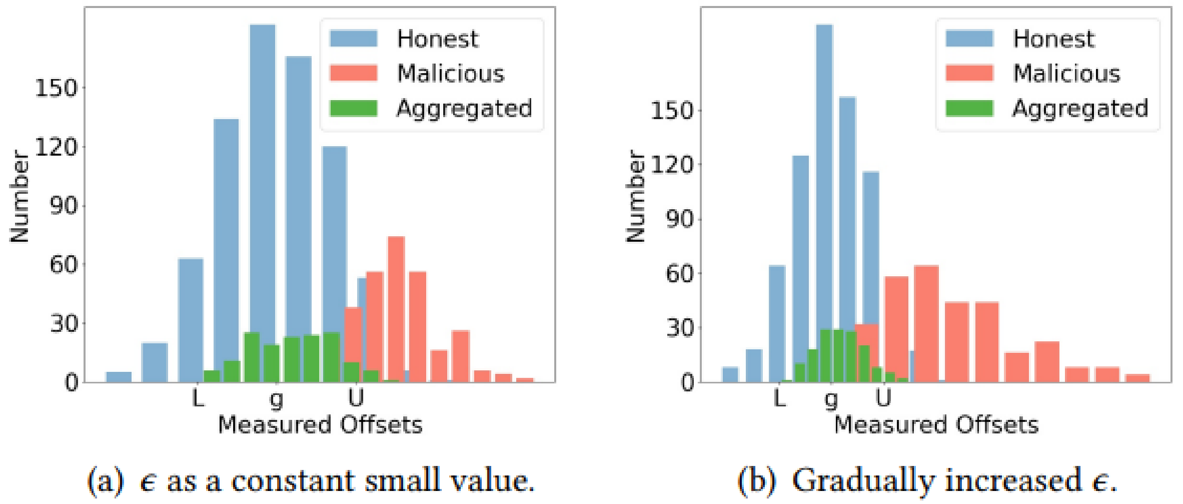
Fig 13: MS-PTP Algorithms

Analysis

- **Byzantine resilience:** We provide mathematical proof for the Byzantine resiliency of MS-PTP. Detail results can be found in our paper.
- **Complexity:** MS-PTP increases the communication overhead by $n \times$ when there are n servers. MS-PTP is a lightweight protocol and does not introduce a lot of computation overhead. The algorithm can be executed within several milliseconds
- **Compatibility:** MS-PTP only requires some customized configurations and is fully compatible with the current PTP standard. In case there are not enough PTP servers in the network, MS-PTP may also resort to alternative external redundancy such as GPS sources and NTP sources.

MS-PTP against Adaptive Attacks

- We analyze MS-PTP's performance against the adaptive attacks assuming they have full knowledge of our defense mechanism with extensive simulations.
- The attacker can conduct the following adaptive attack:
  - Phase 1: Standard Deviation Estimation. The attackers conclude with each other to estimate the standard deviation of the honest measurements as std.
  - Phase 2: Adding Malicious Measurements. The attackers introduce malicious measurements as $\sqrt{2}std + \varepsilon$.
  - ε can be a constant or increasing value.
- They still cannot break MS-PTP!



(a) ϵ as a constant small value. (b) Gradually increased ϵ.

Fig 16: MS-PTP against adaptive attack.

Evaluation-Experiment Settings

- We use **four Raspberry Pis** to emulate the server pools that MS-PTP requires to counter Byzantine failures.
- The PTP client takes the IP address (in LAN) of all four Raspberry Pis and configures its profile to establish independent sessions with them.
- This testbed supports hardware validation when the malicious population f = 1 and the number of servers n = 3f + 1 = 4.
- We open up multiple PTP sessions, each bound to an independent CPU core on the server node to experiment with a larger malicious population and testify to the overhead introduced by MS-PTP. Our testbed suffices for the experiment settings when **n ≤ 16**.

Evaluation-Resiliency Against Attacks

- We evaluated the performance of MS-PTP still on our PTP testbed. We focused on checking the accuracy of the evaluated offsets of the system when the attack was launched.
- MS-PTP is resilient to the insider attack.

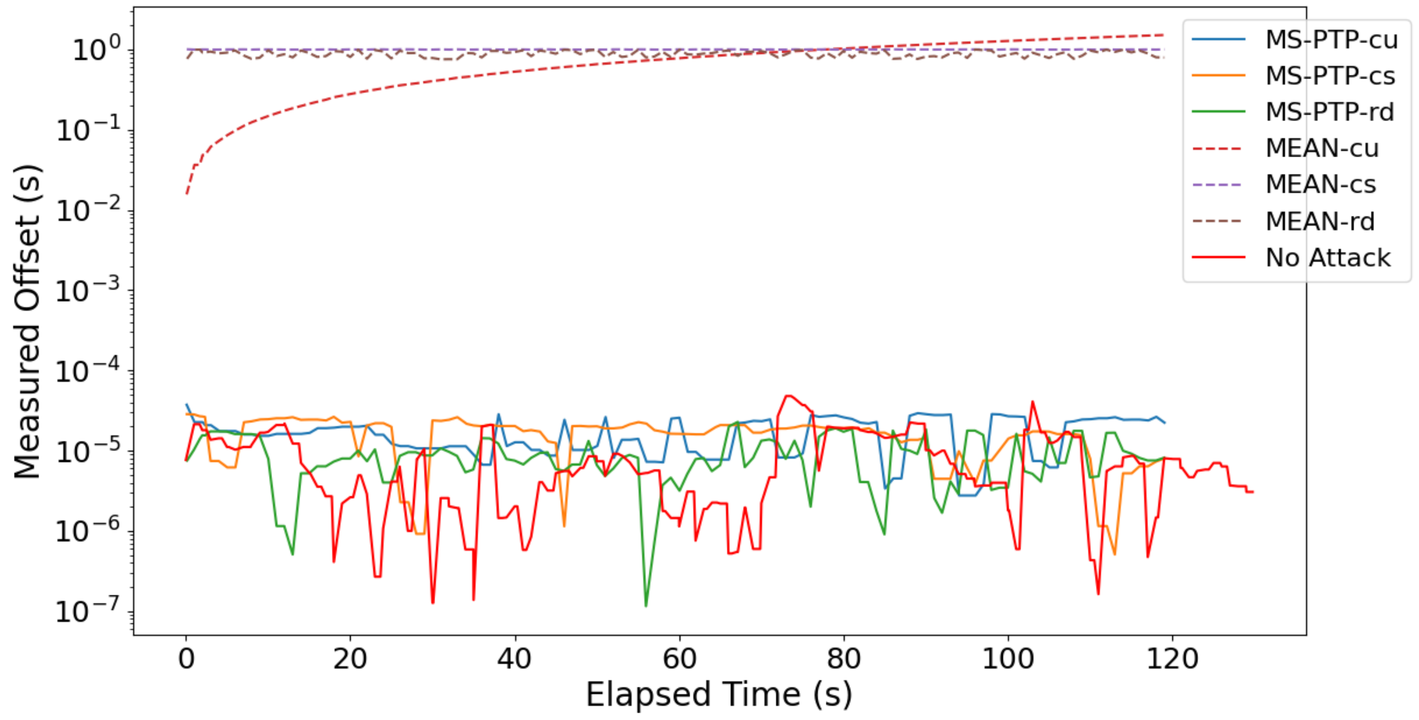
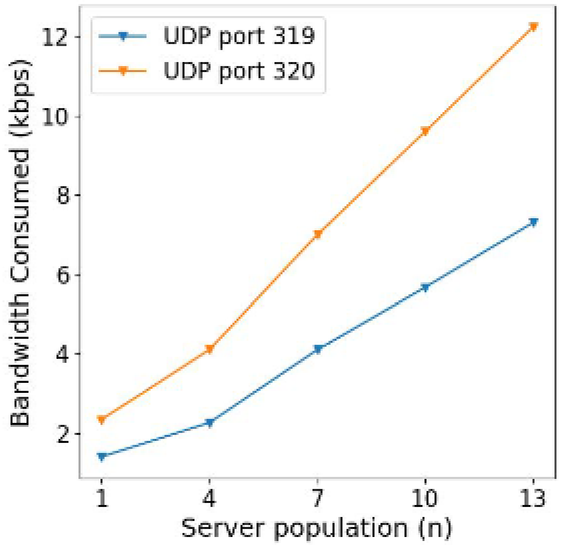


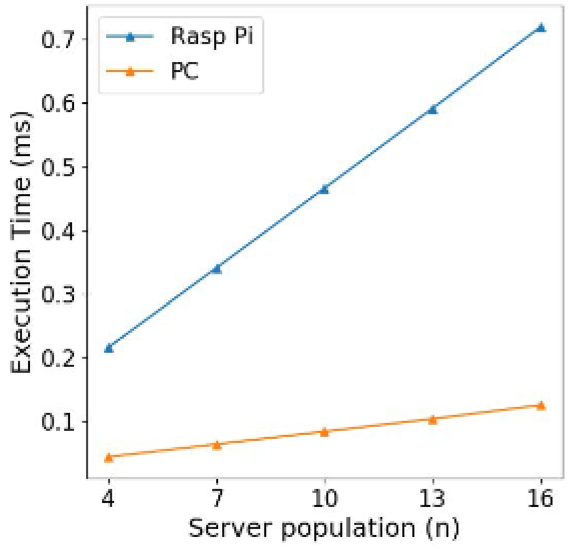
Fig 14: MS-PTP accuracy performance under attack. In the figure, cu refers to cumulative delay, cs refers to constant delay and rd refers to random delay.

Evaluation-Comm and Comp overhead

- For the communication overhead, we monitored the bandwidth consumed by the UDP port 319 and 320 used by PTP during our experiment on the testbed.
- For the computation overhead, we checked MS-PTP's execution time on one PC and Raspberry Pi.
- MS-PTP only introduced a small linear increasing overhead to the system.



(b) MS-PTP communication overhead.

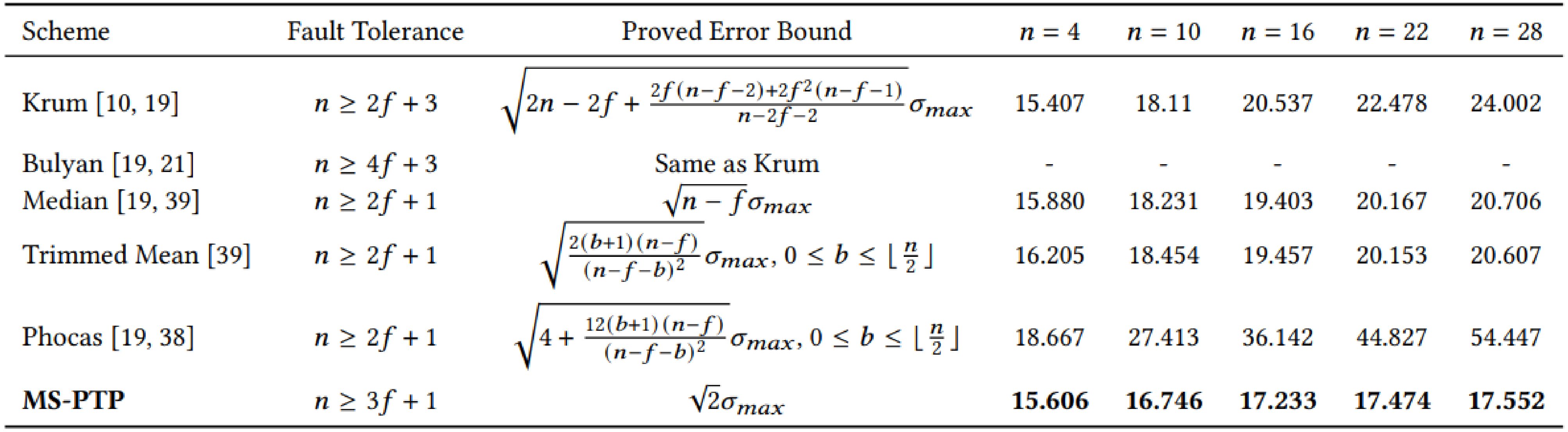


(c) MS-PTP computation overhead.

Fig 15: BRENTS' communication and computation overhead.

Evaluation-Scalability

- We compared MS-PTP with other aggregation methods. The error bound of MS-PTP remained to be stable and was not affected by the population of network devices in our simulations.

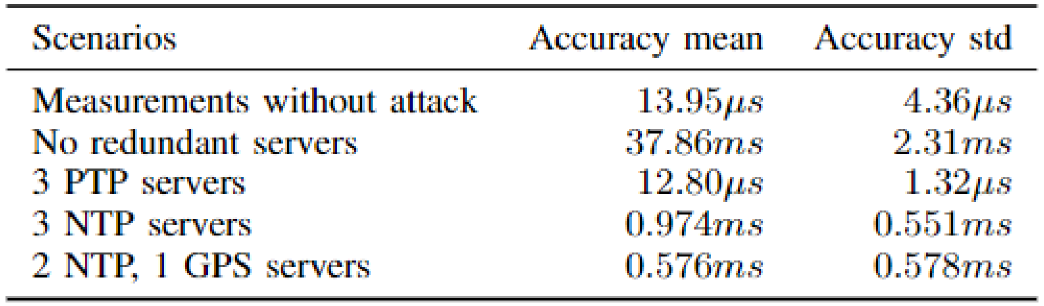


Scheme	Fault Tolerance	Proved Error Bound	$n = 4$	$n = 10$	$n = 16$	$n = 22$	$n = 28$
Krum [10, 19]	$n \geq 2f + 3$	$\sqrt{2n - 2f + \frac{2f(n-f-2)+2f^2(n-f-1)}{n-2f-2}}\sigma_{max}$	15.407	18.11	20.537	22.478	24.002
Bulyan [19, 21]	$n \geq 4f + 3$	Same as Krum	-	-	-	-	-
Median [19, 39]	$n \geq 2f + 1$	$\sqrt{n - f}\sigma_{max}$	15.880	18.231	19.403	20.167	20.706
Trimmed Mean [39]	$n \geq 2f + 1$	$\sqrt{\frac{2(b+1)(n-f)}{(n-f-b)^2}}\sigma_{max}, 0 \leq b \leq \lfloor \frac{n}{2} \rfloor$	16.205	18.454	19.457	20.153	20.607
Phocas [19, 38]	$n \geq 2f + 1$	$\sqrt{4 + \frac{12(b+1)(n-f)}{(n-f-b)^2}}\sigma_{max}, 0 \leq b \leq \lfloor \frac{n}{2} \rfloor$	18.667	27.413	36.142	44.827	54.447
MS-PTP	$n \geq 3f + 1$	$\sqrt{2}\sigma_{max}$	**15.606**	**16.746**	**17.233**	**17.474**	**17.552**

Tab 4: MS-PTP scalability performance (measured offset in microseconds) under attack.

Evaluation-Compatibility

- We validated MS-PTP's compatibility with NTP and GPS-based synchronization methods by introducing NTP servers and GPS servers as redundant time sources.
- We considered the existence of one malicious PTP server, but with either 3 NTP servers or 2 NTP servers plus 1 GPS server serving as the honest redundancy.
- The result shows that MS-PTP is compatible with other time synchronization mechanisms.

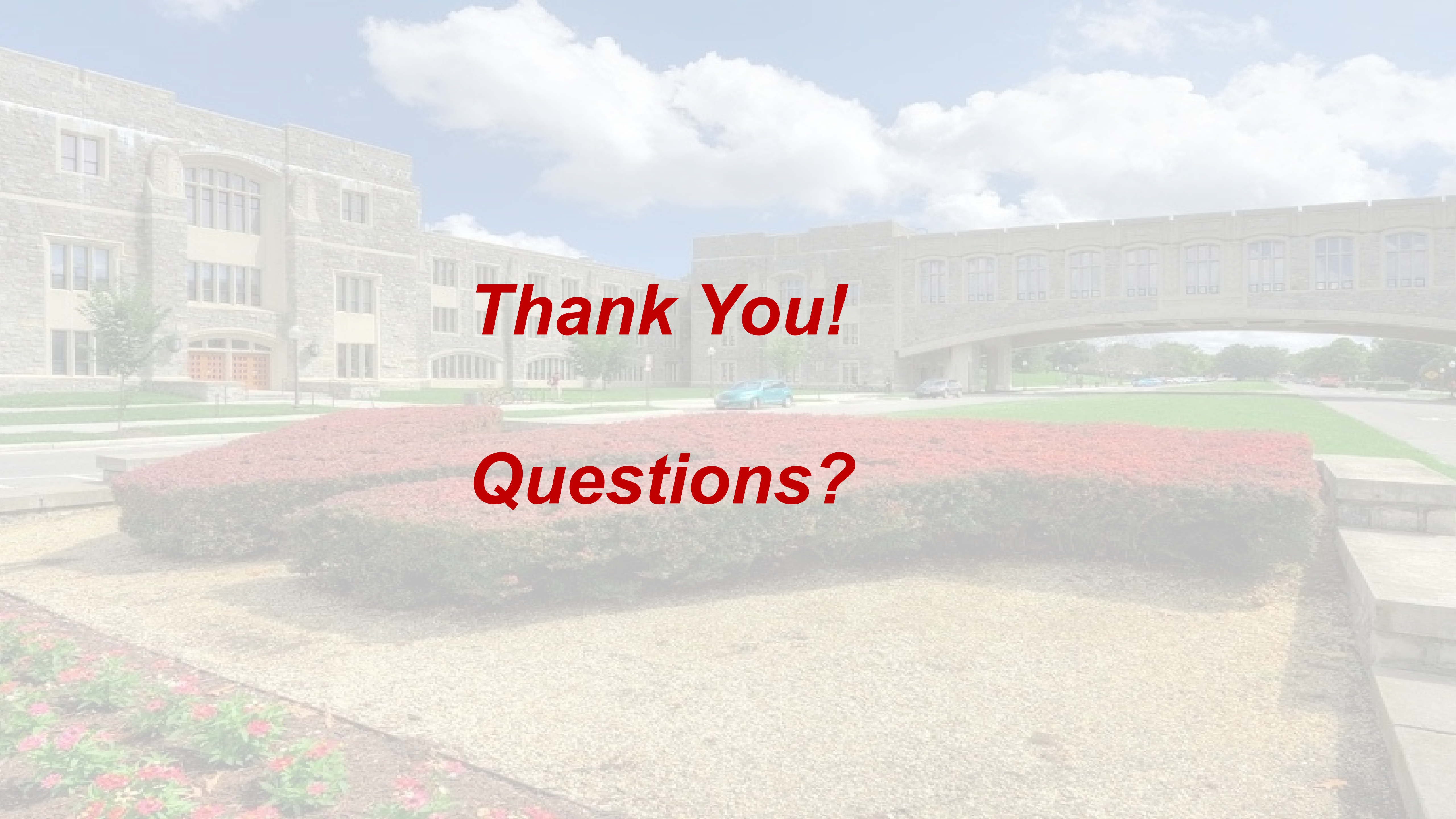


Scenarios	Accuracy mean	Accuracy std
Measurements without attack	$13.95\mu s$	$4.36\mu s$
No redundant servers	$37.86ms$	$2.31ms$
3 PTP servers	$12.80\mu s$	$1.32\mu s$
3 NTP servers	$0.974ms$	$0.551ms$
2 NTP, 1 GPS servers	$0.576ms$	$0.578ms$

Tab 5: MS-PTP's offset accuracy with GPS and NTP servers.

Conclusion

- We proposed MS-PTP, a Byzantine-resilient network time synchronization mechanism, to provide accurate clock synchronization service to clients without relying on the trust of any individual time source.
- We first demonstrated through hardware experiments that the current PTP is susceptible to the insider time-shifting attack, in which one malicious time source can bring catastrophe to the system.
- We devised a novel Byzantine-resilient aggregation scheme to generate a high-fidelity, error-bounded measurement under the assumption that fewer than one-third of the measurements are Byzantine-influenced.
- We implemented a proof-of-concept MS-PTP with our hardware testbed and evaluated its performance in various Byzantine-ridden network scenarios.



Thank You!

Questions?