Taming Asymmetric Network Delays for Clock Synchronization Using Power Grid Voltage

Dima Rabadi[1,3] Rui Tan[2] David K. Y. Yau[1,3] Sreejaya Viswanathan[3]
[1]Singapore University of Technology and Design [2]Nanyang Technological University, Singapore
[3]Advanced Digital Science Center, Illinois at Singapore

ABSTRACT

Many clock synchronization protocols based on message passing, e.g., the Network Time Protocol (NTP), assume symmetric network delays to estimate the one-way packet transmission time as half of the round-trip time. As a result, asymmetric network delays caused by either network congestion or malicious packet delays can cause significant synchronization errors. This paper exploits sinusoidal voltage signals of an alternating current (ac) power grid to tame the asymmetric network delays for robust and resilient clock synchronization. Our extensive measurements show that the voltage signals at geographically distributed locations in a city are highly synchronized. Leveraging calibrated voltage phases, we develop a new clock synchronization protocol, which we call Grid Time Protocol (GTP), that allows direct measurement of one-way packet transmission times between its slave and master nodes, under an analytic condition that can be easily verified in practice. The direct measurements render GTP resilient against asymmetric network delays under this condition. A prototype implementation of GTP, based on readily available ac/ac transformers and PC-grade sound cards as voltage signal sampling devices, maintains sub-ms synchronization accuracy for two nodes 30 km apart, in the presence of malicious packet delays. We believe that GTP is suitable for grid-connected distributed systems that are currently served by NTP but desire higher resilience against network dynamics and packet delay attacks.

Keywords

Clock synchronization; power grid; security

1. INTRODUCTION

Secure clock synchronization is critical for many mission-critical distributed system applications. For instance, in common SCADA-controlled infrastructures, various computing nodes and intelligent electronic devices (IEDs) in an advanced manufacturing system monitor collaboratively the system state in real time, in order to run a set of remote terminal units (RTUs) and PLC-enabled actuators. In such systems, for safe and effective operation, the computers, IEDs, and actuators must be tightly synchronized, to within a few milliseconds [6].

Many of today's cyber-physical systems have mainly employed the Network Time Protocol (NTP) [2] in the system's local area networks (LANs) to distribute UTC time from GPS-equipped masters to various slaves. More generally, NTP is a foremost means of network time synchronization that is widely known and adopted. Its design principles are also representative of a large class of synchronization protocols based on message exchanges between the synchronizing nodes. Other examples of these protocols include Precision Time Protocol (PTP) [4] and those for wireless sensor networks such as RBS [7], TPSN [8], and FTSP [16].

In normal operation, NTP's accuracy is generally accepted to be within a few milliseconds. However, NTP is susceptible to a number of attacks. Certain of these attacks can be detected by protocol constructs with cryptographic protection. For example, authenticated sequence numbers can guard against malicious dropping of packets, and signed messages or message digests can ensure the integrity of the message content. A simple but powerful form of attack against NTP (or any synchronization protocols based on message passing), which has evaded satisfactory detection and mitigation so far, is malicious packet delays. In this attack, the adversary on a forwarding path of the NTP packets can maliciously delay one direction of the communications between the slave and master. Because the malicious delay does not change the message itself, it is immune to cryptographic protection. It is effective, however, because it invalidates a basic *symmetric link assumption* of NTP [22]. Specifically, a malicious delay of d can introduce a synchronization error up to $\frac{d}{2}$; errors on the order of 10 ms up to seconds are eminently feasible. There are various heuristic approaches to detecting and mitigating the delay attack [22, 21, 18], but none of these are completely foolproof. In Section 3, for example, we will demonstrate a subtle attack via ARP spoofing that can evade detection based on tracking historical round-trip times (RTTs) by moving averages [11, 20, 22].

As NTP cannot measure directly one-way transmission times of its synchronization packets for clock offset calculations, it relies on the symmetric link assumption to estimate the one-way transmission time as half of the RTT in a slave-master communication. In this paper, we seek a trustworthy external signal that both the slave and master can observe, so that they can measure directly the one-way transmission times. A new synchronization approach based on this di-

ASIA CCS '17, April 02-06, 2017, Abu Dhabi, United Arab Emirates
© 2017 ACM. ISBN 978-1-4503-4944-4/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3052973.3053020

rect measurement will no longer need the symmetric link assumption; it will be resilient against packet delay attacks. To realize the approach, we exploit the voltage waveforms of an alternating current (ac) power grid for trustworthy and accurate clock synchronization between distributed network nodes. In this paper, we assume that the power grid voltage signal is intact, because tampering with it often raises large barriers economically and logistically for would-be attackers. The grid's voltage is location dependent; its values at different monitoring points are different. However, in an ac power grid, the sinusoidal voltage waveforms at all the locations are driven by a same frequency. In existing practice, this frequency is either 60Hz (e.g., in the Americas) or 50Hz (in most other parts of the world). Hence, the periodicity of the waveforms is synchronized, although the synchronization is imperfect because the phase of the voltage signal changes with location, and the grid's frequency is not truly constant but it is continuously regulated around the nominal value in response to changes in load and generation. An important research question that we seek to answer is whether this synchronization is good enough for practical applications.

To answer the question, we conduct extensive measurements in a city to verify key synchronization properties of the ac grid voltage. Based on the results, we design and implement a new clock synchronization approach, which we call Grid Time Protocol (GTP), that (i) achieves better accuracy than NTP, and (ii) is resilient against malicious packet delays. Moreover, we achieve an economical design that can be readily and widely adopted by commodity computing devices with direct utility power access. We make the following main contributions in this paper.

- We verify by real-world experiments that a succinct *phase angle* feature of voltage waveforms exhibits suitable range and stability for accurate and trustworthy distributed clock synchronization.
- Based on the phase angle, we design GTP that achieves sub-ms accuracy in both LAN and city-scale wide-area network (WAN) settings. This accuracy represents a significant improvement over that of NTP, whose errors are often reported to be on the order of ms or even tens of ms [24]. Moreover, unlike NTP, GTP is resilient against malicious packet delays subject to an easy-to-verify condition, which we call the *GTP condition*, that is made clear by our analysis.
- We have designed and implemented a working prototype of GTP using PC-class sound cards, general purpose operating system (OS), and a low-cost voltage sensor design. Our experiments demonstrate predominant achievement of the GTP condition under diverse settings, including congested networks in WAN scale. They also demonstrate ready applicability of GTP to nodes that are connected to the same power grid, and verify GTP's accuracy and robustness in both LAN and WAN scales.
- We show that, unlike NTP, GTP achieves unambiguous trustworthy synchronization under the GTP condition. We leverage this property to design a resilience policy for running GTP in practical networks with access to multiple potential GTP masters. The resilience policy ensures trustworthy clock synchronization when some but not all of these masters are under attack.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 analyzes the impact of malicious packet delays on NTP and demonstrates this impact via experiments. Section 4 presents extensive grid voltage measurements to establish the foundation of GTP. Section 5 presents the design, performance analysis, and empirical evaluation of GTP. Section 6 proposes a resilience policy for running GTP in practical networks. Section 7 discusses the limitations of GTP. Section 8 concludes.

2. RELATED WORK

Clock synchronization is a fundamental system function of computer networks. There are two broad categories of clock synchronization approaches based on message passing and external periodic physical signals, respectively. Message passing approaches estimate the clock offset between two network nodes by measuring the RTT and one-way transmission times [12]. In NTP [17], time servers (i.e., masters) are organized into a layered hierarchy, where each layer is called a *stratum* and a smaller stratum number means a layer closer to the groundtruth time sources (e.g., atomic clocks or GPS receivers). A stratum-n master updates its clock according to clock offsets estimated from the RTTs of multiple stratum-n and stratum-(n-1) masters. Various message-passing clock synchronization protocols have also been proposed for wireless sensor networks, such as RBS [7], TPSN [8], and FTSP [16]. As the physical distance between two sensor nodes is often limited, these protocols generally ignore the propagation delays of the radio messages used, but they can still achieve high accuracy due to hardware-level timestamping for the exchanged messages.

Recent work has leveraged various external periodic physical signals to synchronize low-power devices or extract timestamps from recorded data. In [19], Rowe *et al.* propose a hardware device called Syntonistor to sense a periodic electromagnetic signal radiating from powerlines and use it to calibrate[1] the clocks of wireless sensors. In [15], Li *et al.* use light sensors to sense the intensity of a fluorescent light that flickers at a frequency twice that of the ac grid frequency. The periodic flickering is used to calibrate the clocks of nodes. Similarly, other external periodic signals in FM radios [14] and Wi-Fi beacons [10] have been leveraged for clock calibration. Using the above clock synchronization approaches, multiple nodes remain synchronized once they are initially synchronized. The initial synchronization, however, requires the exchange of network messages, which may be subverted by packet delay attacks. However, none of these studies address the packet delay attacks against this initial synchronization, but we do. The fluctuations of power grid frequency provide a fingerprint indicative of time. Garg *et al.* [9] extract a grid frequency trace from video recordings, comprising scenes that contain fluorescent light flickering, to identify the recording time.

Recent research has studied the security of clock synchronization approaches. NTP is susceptible to integrity and packet delay attacks. An integrity attack that modifies data fields in the synchronization packets can be addressed by cryptographic encryption. A packet delay attack adds malicious time delays to the transmissions of NTP synchronization packets, which invalidates the protocol's symmet-

[1] *Clock calibration* ensures that different clocks will advance at the same speed; *clock synchronization* regulates the clocks to have the same value.

ric link assumption. Various heuristic approaches have been proposed to detect packet delay attacks, but none of them can provide complete detection. These approaches include setting an upper bound for allowed RTTs [22], comparing the latest RTT with the RTT history [21], and comparing the RTT of NTP with those of other protocols [18]. These heuristic detectors can be bypassed by small attack delays, gradually increased delays, and delays added to all the packets of a victim node. Although more stringent detection thresholds can be used to limit the attack's impact, they will lead to high false alarm rates under dynamic network conditions. This observation will be demonstrated via experiments in Section 3. In contrast, the GTP proposed in this paper exploits an ac electric grid's periodic voltage signal to measure directly one-way packet transmission times between the slave and master. This approach fundamentally decouples GTP from the symmetric link assumption, and renders it immune to packet delay attacks.

The IRIG-B time code standard has been widely used in industry for distributing time information. However, the IRIG-B-based time distribution systems are generally based on a dedicated non-IP network that needs extra cabling. In contrast, GTP is based on IP networks, which well meets the need of the proliferating IP-based Industrial Internet of Things (IoT) devices.

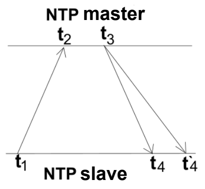
3. MOTIVATION

NTP is the most widely adopted clock synchronization protocol in computer networks. Its design is representative of a large class of the protocols based on message passing. This section reviews NTP and analyzes the impact of an *asymmetric delay attack* on its performance. We will demonstrate respectively NTP's performance in normal operation and under the attack, including a subtle attack designed to overcome an existing moving average based attack detector.

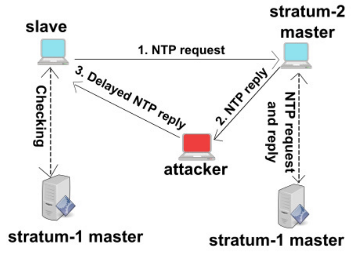
3.1 Impact of Packet Delay Attack on NTP

3.1.1 NTP Principle

As described in Section 2, the nodes running NTP are organized into a layered hierarchy. Each node often runs as both slave and master. For instance, a stratum-n node acts as a slave in synchronizing itself with a stratum-(n-1) node, and as a master when providing its clock values to a stratum-(n+1) node or other stratum-n nodes. Let us consider a pair of NTP master and slave. Fig. 1(a) illustrates a synchronization session between them. The slave starts by sending a request that contains the time of sending the request based on the slave's clock (t_1); the master receives the request and sends back a reply that contains the time of receiving the NTP request from the slave (t_2) and the time at which this reply is sent (t_3), where both t_2 and t_3 are according to the master's clock. When the slave receives the reply, it records the receive time (t_4) and then computes the offset (Δ) between its and the master's clocks, based on the RTT computed from the quadruple time values ($t_1, t_2, t_3,$ and t_4). The RTT and offset are calculated as $\mathrm{RTT} = (t_4 - t_1) - (t_3 - t_2)$ and $\Delta = t_3 + \frac{\mathrm{RTT}}{2} - t_4$. The above offset calculation is based on a *symmetric link assumption*, i.e., the one-way delays for transmitting the request and the corresponding reply are equal. Based on the computed offset, the slave will calibrate its clock.



(a) An NTP session.



(b) Experimental network.

Figure 1: NTP in normal operation and under the asymmetric delay attack, and experimental network.

3.1.2 Asymmetric Delay Attack

Asymmetric links in practice will lead to synchronization errors in NTP, e.g., when an attacker introduces malicious time delays in transmitting either the request or reply packet. We now formally define the threat model of asymmetric delay attack as follows.

Threat model: We assume that the endpoints (master and slave) of a clock synchronization protocol are trustworthy. However, one or more attackers on a network path of the protocol's packets may delay the transmission of these packets. We assume that the total malicious delay for a packet is finite. Moreover, we assume that the protocol's packets cannot be tampered with because of cryptographic protection.

We now analyze the impact of the asymmetric delay attack on NTP. Fig. 1(a) illustrates a case in which the attacker delays the slave's receive of the NTP reply from t_4 to t_4', which we assume to be still within the NTP's default timeout (normally 1 s [1]).[2] The malicious delay will compromise the offset computation. The computed offset under attack is given by $\Delta' = t_3 + \frac{\mathrm{RTT}}{2} - t_4' = t_3 + \frac{t_4' - t_1 - (t_3 - t_2)}{2} - t_4'$. Thus, the added offset is $\Delta' - \Delta = \frac{t_4 - t_4'}{2}$.

Note that, in general, the attacker needs to attack only one direction of the communications, because a delay in the other direction would mitigate the effects of the first. Specifically, if the attacker delays the master's receive of the NTP request from t_2 to t_2' and the slave's receive of the NTP reply from t_4 to t_4', the offset computed by the slave, denoted by Δ'', is $\Delta'' = t_3 + \frac{\mathrm{RTT}}{2} - t_4' = t_3 + \frac{t_4' - t_1 - (t_3 - t_2')}{2} - t_4'$. Thus, the added offset is $\Delta'' - \Delta = \frac{(t_4 - t_4') - (t_2 - t_2')}{2}$. We can see that, if the attacker introduces the same delay to the request and reply packets (i.e., $t_4 - t_4' = t_2 - t_2'$), the attack has no effect (i.e., $\Delta'' = \Delta$). Delaying one direction of the communications is the most effective attack.

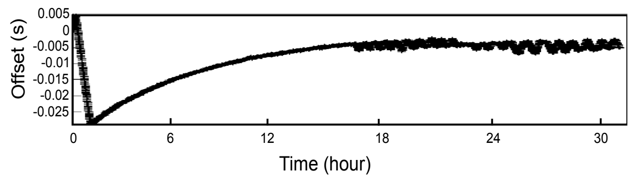
3.2 Two Asymmetric Delay Attack Experiments

This section presents two experiments to demonstrate the actual impact of asymmetric delay attacks on NTP.

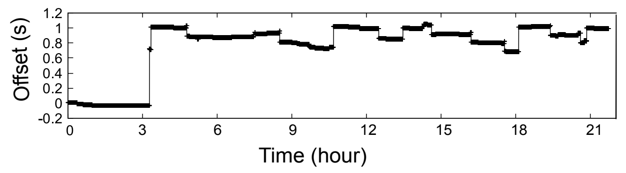
3.2.1 Experiment Setup

The experiment setup is shown in Fig. 1(b). The setup consists of three computers in the same LAN acting as the

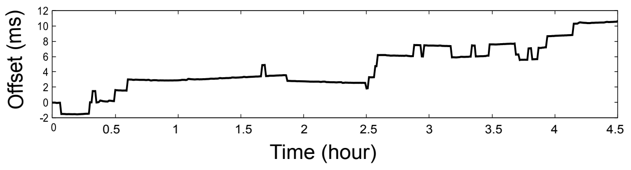
[2]Replies received after the timeout will be discarded, and the slave will resend its request to the master.



(a) No asymmetric delay attack.



(b) Asymmetric delay attack starts from the 3rd hour.



(c) Gradually increasing delay attack.

Figure 2: Slave's clock offset from GPS time.

NTP slave, (stratum-2) NTP master, and attacker, respectively. The NTP daemon `ntpd` running on the slave and master are from the NTP reference implementation `ntp-4.2.8p3` [2]. The NTP slave is configured to synchronize with the NTP master, which synchronizes with higher-level (stratum-1) NTP masters equipped with GPS receivers. The slave's `ntpd` daemon periodically initiates NTP sessions illustrated in Fig. 1(a), based on which it calibrates the slave's clock. To assess the synchronization error, the slave uses the `ntpdate` utility to periodically check its time offsets against the stratum-1 masters with groundtruth GPS time.

Asymmetric delay attacks on the NTP reply messages are via Address Resolution Protocol (ARP) spoofing, which is practical because ARP generally must be enabled for a LAN's normal operation. The spoofing allows the attacker to intercept the NTP replies. Then, using a traffic control utility `tc`, the attacker delays each reply for a specified amount of time before forwarding it intact to the slave.

3.2.2 Asymmetric Delay Attack with Fixed Delay

We first report NTP's synchronization errors in the absence of attacks. Fig. 2(a) profiles the offsets of the slave's clock from the stratum-1 time masters over time. We can see that the offsets converge to approximately 5 ms, which is a typical synchronization error achieved by NTP in LAN settings. In the second experiment, the slave uses Apache's Java implementation of NTP (`org.apache.commons.net.ntp`) to synchronize with the master. By default, the timeout of this NTP implementation is not enabled. The attacker introduces a malicious delay of 2 s to the NTP reply packets. As shown in Fig. 2(b), the measured offsets are around 1 s, which is half of the malicious delay and consistent with our analysis in Section 3.1.

3.2.3 Bypass Moving Average based Detection

Section 2 reviews state-of-the-art methods for detecting packet delay attacks. A widely adopted approach [11, 20, 22] tracks the average RTT of protocol packets (NTP packets in our context) over time windows, and checks that any changes of the average between consecutive windows fall within a predetermined threshold. For example, if the window size is four, the average is calculated based on the last four RTT samples. This average is updated every time a new NTP session occurs and a new RTT sample hence becomes available. If the difference between the new and old averages exceeds the threshold, the detection declares an attack. The detection guards against abrupt increases in the RTT as evidence of attacks. Thus, to avoid detection, the attacker must limit the malicious delay to a modest value, which in turn limits the clock drift that the attacker can cause.

Unfortunately, a practical attacker can launch a series of small attacks whose delay increments stay below the detection threshold but add up to a significant cumulative delay over time. To verify its effectiveness, we launch such a gradually increasing delay attack on the NTP using the same ARP spoofing mechanism as before. The window size is four and the detection threshold is 3 ms. Before the attack starts, the last four measured RTTs are 2, 2, 2, and 2.5 ms, respectively, for a logged average of 2.125 ms.

The attacker starts with a small malicious delay of 2 ms, resulting in an observed new RTT sample of 4 ms and, accordingly, a new average RTT of 2.625 ms. The moving average increases by 0.5 ms, well within the threshold, and the clock drifts maliciously by 1 ms, which is half the delay as analyzed in Section 3.1.2. Then, the attacker gradually increases the delay from 2 ms to 20 ms. Fig. 2(c) tracks the offset of the slave's clock from the groundtruth time after each instance of the malicious delay. Note that the achieved offset increases to 10 ms, as shown in Fig. 2(c), while the average RTT differences keep below the detection threshold.

3.3 Implication on Time-critical Systems

NTP is widely used in various industrial systems. For instance, from our communication with power plant operators and power grid SCADA system integrators, NTP remains the main method used to synchronize various computers and intelligent electronic devices (IEDs) in a power substation with a GPS-based time master. These computers and IEDs monitor the status of power generators and the grid, and control various critical actuators. Because of fast dynamics of electricity, the asymmetric delay attack poses a real danger to the time-critical generator/grid sensing and control. Such an attack can be launched in the LAN of a power substation by, for example, an insider attacker or a worm that has bypassed the system's air gap.

4. POWER GRID VOLTAGE PROFILES

This paper exploits the power grid voltage to tame the asymmetric network delays for message passing-based clock synchronization. This section presents the background for power grid voltage and our extensive measurements in a city-scale power grid to provide a design basis for the proposed clock synchronization approach.

4.1 Background

Alternating current (ac) power grids are designed to run at a prescribed nominal frequency (60 Hz in the Americas and 50 Hz in most other places of the world) [13]. This frequency, which is the frequency of the sinusoidal ac voltage, is almost

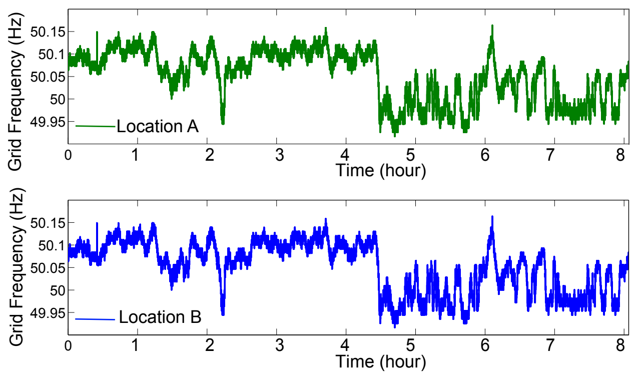


Figure 3: Power grid frequency measured at two locations 30 km apart.

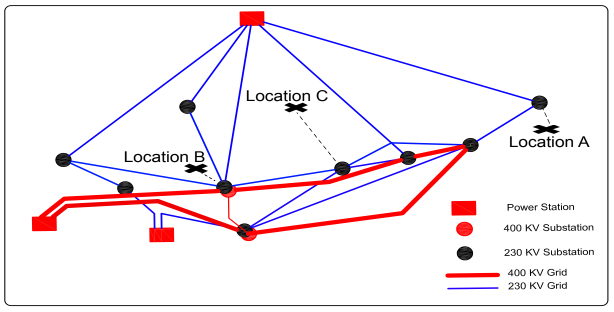


Figure 4: The locations of three distributed nodes with respect to the transmission system of our city. The distance between nodes A and B is 30 km; that between nodes B and C is 15 km. A dashed line connects a location with its transmission system bus.

identical across all locations in a grid [13]. Because of constant load dynamics, however, the frequency is not fixed but it must be continuously regulated around the nominal value by, for example, dynamic generation control. This causes persistent albeit small fluctuations of the frequency around the nominal value over time, as Fig. 3 shows for our frequency measurements at two separate locations 30 km apart in our city. Previous work [23] uses these frequency fluctuation signatures that are indicative of time information for synchronizing distributed devices. However, to capture the often minute fluctuations, the prior approach [23] employs a highly customized hardware peripheral to achieve high-rate and high-precision sampling, which increases the system cost.

In this paper, we take an alternate economical approach of analyzing the angle of the periodic ac voltage signals. Specifically, the normalized power grid voltage at a location l, denoted by $v_l(t)$, is given by $v_l(t) = \sin(2\pi f_l(t)\cdot t + \psi_l(t) + \phi_l)$, where t is the global time, $f_l(t)$ is the localized grid frequency as shown in Fig. 3, $\psi_l(t)$ is the voltage angle, and ϕ_l denotes the grid phase that will be explained shortly. As shown in Fig. 3, $f_l(t)$ changes over time due to transient imbalance between active power generation and active load. It can be slightly different across different locations. The voltage angle often has a small value and changes over time due to the changing geographic distribution of active load in a power grid [13]. Most existing power grids have three phases, i.e., the ϕ_l takes on three possible values of 0, $-2\pi/3$, and $2\pi/3$. In a building, the wall power outlets are often evenly distributed among these three phases, depending on the configuration of the power distribution network. To simplify discussions, we assume that the synchronizing slave and master at two locations l_c and l_s, respectively, observe the same power grid phase, i.e., $\phi_{l_c} = \phi_{l_s}$. In the experiments conducted in this paper, we select the wall power outlets to match this assumption. In Section 5.4, we will discuss how to address the case in which the slave and master observe different power grid phases.

In this paper, the angle of v_l is defined as $\theta_l(t) = 2\pi f_l(t)\cdot t + \psi_l(t)$, which will be the basis of our new clock synchronization protocol. To use $\theta_l(t)$ for the synchronization, our main concern is whether the difference between the voltage angles of the slave and master (in radians), i.e., $\theta_{l_c}(t) - \theta_{l_s}(t)$, remains near-constant despite the changes of f_l and ψ_l. To simplify discussion, the *angle difference* between the slave and the master in milliseconds is defined as

$$\gamma(t) = \frac{\theta_{l_c}(t) - \theta_{l_s}(t)}{2\pi}\cdot p, \tag{1}$$

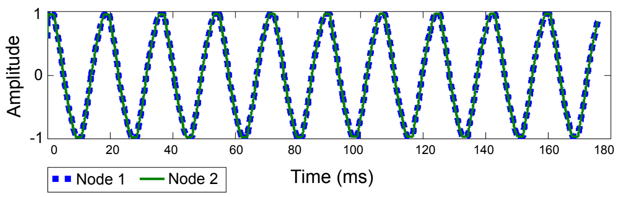
where p is the nominal ac cycle time duration, e.g., $p = 20$ ms for a 50 Hz power grid. Thus, the $\gamma(t)$ captures the impact of the active load fluctuation and its varying geographic distribution over time. If the angle difference is unknown, it will be part of the synchronization error of the proposed approach; otherwise, we can exclude it from the calculations. As a result, the range and stability of this angle difference are important. A stable angle difference will allow us to calibrate the proposed system using the average value of $\gamma(t)$, denoted by $\bar{\gamma}$. A small $\gamma(t)$ makes sure that the synchronization error is small, when $\bar{\gamma}$ cannot be measured for the system calibration.[3] We note that, as ac voltage is cyclic, the voltage angle difference is within $(-\pi, \pi)$ in radians, which is $(-p/2\,\text{ms}, p/2\,\text{ms})$ in time. Thus, in a 50 Hz grid, the upper bound of synchronization error is 10 ms when the GTP system is not calibrated.

From power engineering [13], the transient imbalance between reactive power generation and reactive load causes voltage amplitude fluctuations. GTP is insensitive to the voltage amplitude fluctuations. We will present the details in Section 5.2.1. A load that is not purely resistive will increase the angle between voltage and current (i.e., deteriorate the power factor), which is different from the voltage angle difference between two locations that is of concern in this paper. Thus, the power factor does not affect GTP.

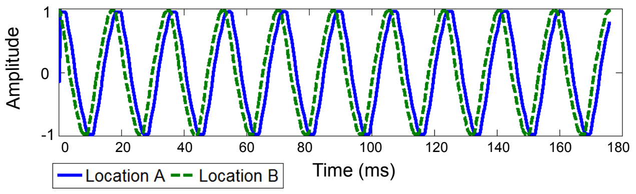
4.2 Power Grid Voltage Measurements

In this section, we capture real ac powerline voltage signals to illustrate their synchronization property. In particular, we analyze the angle difference $\gamma(t)$ between two different locations in a LAN and WAN setting, respectively. In the LAN setting, we use two nodes located on the same floor of a building. For the WAN setting, we use two nodes that are respectively about 15 km (nodes B and C) and 30 km (nodes A and B) apart in our city, as shown in Fig. 4. Note that nodes A, B, and C are within a university, an office building, and a residential building, respectively. The city's grid

[3]The period of such system calibration can be chosen to achieve a satisfactory tradeoff between GTP's synchronization error and overhead introduced by the calibration.

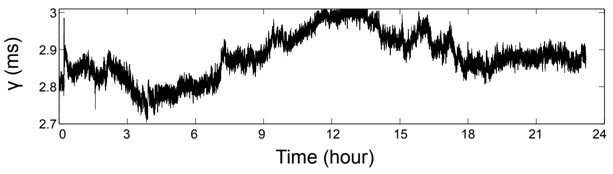


(a) In the same building floor at Location B

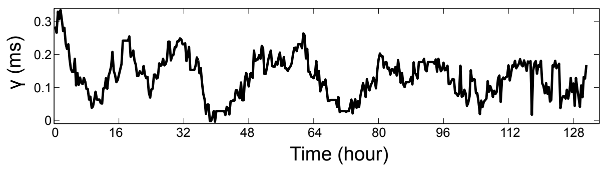


(b) 30 km apart.

Figure 5: Voltage signals measured by two nodes.



(a) 30 km apart over one day. Mean and standard deviation are 2.879 ms and 0.0609 ms, respectively.



(b) 15 km apart over five day. Mean and standard deviation are 0.1324 ms and 0.0628 ms, respectively.

Figure 6: Angle difference between two locations.

frequency is 50 Hz. The measurements are conducted using custom hardware that can capture the instantaneous grid voltage signal at a high sampling rate. This hardware is used for the background study in this section only, whereas our new clock synchronization approach will use another simple hardware setup described in Section 5.1. The two nodes used in this section are (almost) perfectly synchronized in time using GPS receivers placed at their respective locations. The collected voltage signals are aligned based on their GPS timestamps. As our measurement study presented in this section is conducted in a real-world power grid, it addresses the impact of the inevitable active load fluctuation and its varying geographic distribution over time.

4.2.1 *LAN-scale Results*

The voltage profiles of the two nodes in the same LAN are captured at the same time. They are sinusoidal, and Fig. 5(a) shows their data for ten sample ac cycles among a trace that spans one whole day. For the whole day, the $\gamma(t)$ has an average value of $\bar{\gamma} = 0.009$ ms and a standard deviation of 0.0032 ms. Thus, the angle difference is small and stable. Hence, the two voltage signals are almost perfectly synchronized with each other, as shown in Fig. 5(a).

4.2.2 *WAN-scale Results*

We now compare the voltage profiles of two nodes that are significantly apart in the city. In the first experiment, the two nodes are located at locations A and B that are about 30 km apart. The voltage signals at the two nodes for a sample of ten ac cycles are shown in Fig. 5(b). Notice that, compared with the previous LAN setting, the angle difference between the signals becomes noticeable. Since the angle difference can be affected by changing load distribution of the grid, we ascertain its stability throughout one day, which encompasses say both high load during daytime and low load during late night, as well as concomitant load redistributions. Fig. 6(a) profiles the angle difference over time, with an average value of $\bar{\gamma} = 2.8$ ms and a standard deviation of 0.0609 ms. This shows that the angle difference is small and stable.

We report a further experiment for two nodes deployed at locations B and C that are 15 km apart, as shown in Fig. 4. Fig. 6(b) shows the angle difference over five days, with an average of $\bar{\gamma} = 0.13$ ms and a standard deviation of 0.0628 ms. Thus, the standard deviation is similar to that obtained for locations A and B.

4.2.3 *Discussion*

A comparison between the LAN setting (Section 4.2.1) and the 15 km and 30 km WAN settings (Section 4.2.2) verifies that the angle difference increases with distance, which is consistent with intuition and knowledge of power engineering [13]. Moreover, the standard deviation of the angle difference remains small (i.e., less than 0.07 ms) over a longer trace that covers different days of the week (specifically, five days that include the weekend), which gives further confirmation that the angle difference is stable.

5. GRID TIME PROTOCOL

We propose a new clock synchronization protocol, Grid Time Protocol (GTP), that utilizes an ac power grid's voltage as a reliable and extrinsic periodic signal to measure asymmetric delays individually, thereby achieving resilient clock synchronization between a master and slave connected to the same grid. Section 5.1 presents a method to capture the ac voltage signal. The core design of GTP, including formal derivations of the crucial one-way delays, is presented in Section 5.2. Section 5.3 presents comparative performance measurements of GTP and NTP in the presence of asymmetric delay attack.

5.1 Voltage Signal Capture

GTP works by leveraging the power grid voltage, a highly accessible and reliable reference signal that is impractical for the adversary to compromise. As Section 4.2 shows, between two geographically distant locations within the same grid, the voltage oscillates uniformly with a near-constant angle difference. Thus, by observing the angle of the grid voltage when a GTP slave/master sends/receives a synchronization packet, we can accurately estimate the one-way packet transmission delays between the two nodes in both directions.

5.1.1 *GTP Hardware*

Our reference implementation of GTP uses the hardware design shown in Fig. 7. GTP analyzes the sinusoidal voltage

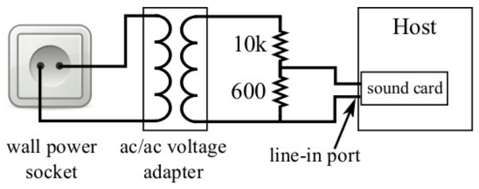


Figure 7: Device for capturing powerline voltage.

signals captured by a commodity PC's sound card. A simple but key hardware device, for both the GTP slave and master, is thus a voltage sensor capable of capturing the subject voltage signal accurately. Off-the-shelf ac/ac voltage adapters can be used as the voltage sensor. However, the output voltages of most ac/ac adapters are higher than the range of the line-in input of PC's sound card. Thus, a voltage divider is needed to interface the ac/ac adapter and the sound card. In our hardware shown in Fig. 7, the voltage divider reduces the peak-to-peak voltage of 17 V given by the ac/ac adapter to 1 V for the sound card. A software application module written in C++ reads the line-in port data from the sound card's driver, which is sampled at 44 kHz in real time. Specifically, the driver continuously samples the line-in signal, and returns a block of data at one-second intervals to the application.

The prototype hardware device shown in Fig. 7 is mainly for commodity desktop and single-board computers. Using sound card as the sampling device has two salient advantages. First, computers are generally equipped with sound cards and their operating systems already provide unified access interfaces. This ensures that GTP is highly portable. Second, as analyzed in Section 5.2.2, the voltage sampling rate is an important factor in GTP's synchronization error, and sound cards provide a sufficiently high sampling rate for small errors. In contrast, customized sampling devices of comparable sampling rates are often expensive. Single-board computers, e.g., Raspberry Pi and BeagleBone boards, often have add-on and built-in analog-to-digital converters to sample the voltage sensor. For battery-powered nodes without direct access to the power grid, we can use a circuit [19] to sense wirelessly powerline electromagnetic radiations, which are highly correlated with the ac voltage signal.

5.1.2 Security of Voltage Signal

Although the threat model considered in this paper is the asymmetric delay attack defined in Section 3.1.2, this section discusses the security of the voltage signal also, since GTP additionally uses this signal. Tampering with power grid voltage signal often raises large barriers economically and logistically for would-be attackers. We discuss the following three possible cases where the attacker may affect the voltage signal.

First, the attacker may inject high-frequency noises, similar to signals generated by power-line communication devices, into related power lines to introduce tiny voltage waveform fluctuations. However, such high-frequency noises can be removed readily by adding a low-pass filter after the voltage divider in Fig. 7.

Second, by manipulating the geographic distribution of load of a power grid (e.g., by disconnecting/connecting a sufficiently large load to the grid), the attacker may affect the voltage angle difference between two remote locations in the grid. However, this attack would be extremely challenging

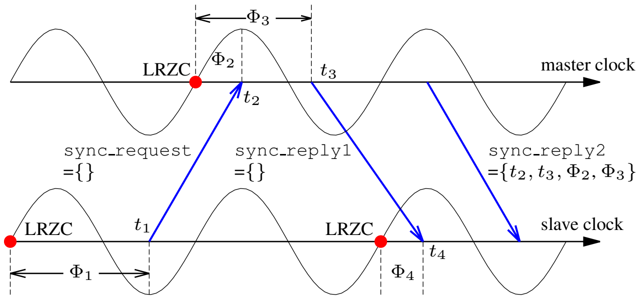


Figure 8: Illustration of GTP operations. The lower and upper sinusoids represent the voltage signals captured by the slave and the master, respectively. The Φ_2 and Φ_3 are computed after `sync_reply1` is sent. The Φ_1 and Φ_4 can be computed after `sync_reply2` is received.

economically and logistically, but can only cause a maximum synchronization error of 10 ms in a 50 Hz grid, because its effect is similar to a poor calibration for the GTP system. In fact, if the attacker is strong enough to obtain the required physical access, disconnecting the targeted area from the utility grid will likely be more attractive to the attacker than compromising clock synchronization, as the attack will leave a significant physical footprint immediately.

Third, the attacker may connect an RC circuit in series to a power distribution line to introduce additional voltage angle difference and affect the accuracy of GTP. This attack requires physical access to the power line, which is logistically difficult. Moreover, its effect is also similar to a poor calibration to GTP, which would incur a maximum synchronization error of 10 ms only. Moreover, if the attacker had gained physical access to the power line, simply cutting it would be a more effective attack.

Given the above considerations, in this paper we assume that the voltage signal is intact.

5.2 Principle of GTP

This section presents the working principles of GTP.

5.2.1 GTP Operations

The basic operations of GTP are illustrated in Fig. 8. In the figure, the sinusoids in the bottom and upper halves represent the voltage signals captured respectively by the slave and master. A synchronization session is initiated periodically or on an on-demand basis. It consists of the transmissions of a request packet and two reply packets. Specifically, the GTP slave program transmits a `sync_request` packet to the master, and records the slave's clock value t_1 when the packet is transmitted. When the master receives the `sync_request`, it records its current clock value t_2 and constructs a `sync_reply1` packet. The master records its clock value t_3 when the `sync_reply1` is transmitted. When the slave receives the `sync_reply1` packet, it records its current clock value t_4. After the master has transmitted the `sync_reply1`, the master program identifies the voltage samples that correspond to the *last rising zero crossings* (LRZCs) of the voltage signal prior to the time instants t_2 and t_3, respectively. For instance, in Fig. 8, the time instants t_2 and t_3 share the same LRZC. Note that, if a new voltage cycle starts between t_2 and t_3, these two time instants will have different LRZCs. Then, the master program computes the elapsed times from t_2's LRZC to t_2 and t_3's LRZC to t_3, which are denoted by Φ_2 and Φ_3, respectively. After that, the master program constructs a `sync_reply2` packet containing t_2, t_3, Φ_2, and Φ_3 and transmits it to the slave. After receiving the `sync_reply2`, as illustrated in Fig. 8, the slave program identifies the LRZCs prior to the time instants t_1 and t_4, and the corresponding elapsed times Φ_1 and Φ_4. Finally, the slave program uses the approach presented in Section 5.2.2 to compute its clock offset from the master. Algorithms 1 and 2 in Appendix A give the pseudocode of the slave and master programs.

We now discuss several important design and implementation considerations:

1. We use the LRZC as the reference point of the elapsed time calculation because it is a salient feature point that can be easily identified. Moreover, as the voltage amplitude fluctuations caused by varying reactive loads of a power grid have little (if any) impact on the positions of zero crossing points, GTP is insensitive to the changes of reactive loads. Note that Φ_1, Φ_2, Φ_3, and Φ_4 are obtained by counting voltage samples – they also do not depend on the voltage amplitude.
2. If we ignore the angle difference between the slave and master, the zero crossings of their signals are synchronized. Our analysis in Section 5.2.2 addresses the angle difference when computing the slave's clock offset.
3. As detailed in Section 5, the voltage signal is sampled at 44 kHz by a sound card. Like many sampling devices, the sound card's driver returns data block by block, where each block has 44 K samples. Thus, to identify the LRZC of any given clock value t, the slave/master program needs to wait until the data block covering t is available. This is implemented by Lines 13 and 6 in Algorithms 1 and 2, respectively.
4. As we will analyze in Section 5.2.2, GTP uses the packets `sync_request` and `sync_reply1` to measure the slave's clock offset, while the `sync_reply2` is an auxiliary packet to convey the timestamps t_2, t_3 and measurements Φ_2, Φ_3. With this auxiliary packet, we can decouple the tasks of timestamping the sending time instant of `sync_reply1` and the signal processing time of computing Φ_3. As a result, the signal processing will not impact the packet receive and send timestamps. Moreover, since we minimize the payload size of the `sync_reply1` packet by deferring the transmission of t_2 to `sync_reply2`, we minimize `sync_reply1`'s transmission delay. This helps GTP to meet the key condition that we will provide in Section 5.2.2 to accurately estimate the slave's clock offset.
5. To simplify the slave program design, we postpone the signal processing of computing Φ_1 and Φ_4 until `sync_reply2` is received. Note that, from our measurements, locating the LRZC and computing the elapsed time Φ takes about 2 ms only on commodity computers. We conclude that the voltage signal processing imposes little overhead.

5.2.2 Clock Offset Analysis

This section analyzes the offset between the slave's and master's clocks based on $\{t_1, t_2, t_3, t_4\}$ and $\{\Phi_1, \Phi_2, \Phi_3, \Phi_4\}$ that are recorded and measured during a GTP synchronization session illustrated in Fig. 8.

We define Θ_1 and Θ_2 by

$$\Theta_1 = \begin{cases} \Phi_2 - \Phi_1, & \text{if } \Phi_2 - \Phi_1 \geq 0; \\ \Phi_2 - \Phi_1 + p, & \text{otherwise.} \end{cases} \tag{2}$$

$$\Theta_2 = \begin{cases} \Phi_4 - \Phi_3, & \text{if } \Phi_4 - \Phi_3 \geq 0; \\ \Phi_4 - \Phi_3 + p, & \text{otherwise.} \end{cases} \tag{3}$$

When a new ac cycle starts during the transmission of the `sync_request` or the `sync_reply1` packet, the $\Phi_2 - \Phi_1$ or $\Phi_4 - \Phi_3$ can be negative, respectively. In this case, we add one ac cycle time duration p, as given in Eqs. (2) and (3). From the descriptions in Section 5.2.1, Φ_i is the elapsed time from the LRZC to some voltage sample. Thus, we have $0 \leq \Phi_i < p$. From Eqs. (2) and (3), we can verify that

$$0 \leq \Theta_1 < p, \quad 0 \leq \Theta_2 < p. \tag{4}$$

The one-way time delays for transmitting `sync_request` and `sync_reply1` can be longer than one ac cycle. To account for this possibility, we use i to denote the non-negative integer number of ac cycles elapsed since the time of sending `sync_request` until the time of receiving it at the master, and j to denote the non-negative integer number of ac cycles elapsed since the time of sending `sync_reply1` until the time of receiving it at the slave. Moreover, as discussed in Section 4.2, a voltage angle difference γ exists between the slave's and master's observations of the signal. From the definition of angle difference in Eq. (1), a positive angle difference means that the slave's voltage signal leads the master's. Denoting by τ_1 and τ_2 the *actual* one-way delays of transmitting the `sync_request` and `sync_reply1` packets, respectively, we have $\tau_1 = \Theta_1 + i \cdot p + \gamma$ and $\tau_2 = \Theta_2 + j \cdot p - \gamma$. Moreover, the RTT computed by $\text{RTT} = (t_4 - t_1) - (t_3 - t_2)$ must satisfy

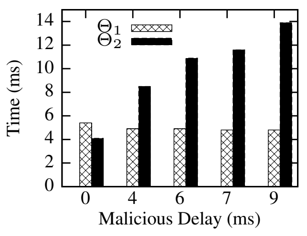
$$\text{RTT} = \tau_1 + \tau_2 = \Theta_1 + \Theta_2 + (i + j) \cdot p. \tag{5}$$

We have the following proposition.

PROPOSITION 1. $\text{RTT} = \Theta_1 + \Theta_2$ *is a sufficient and necessary condition for GTP to unambiguously estimate the two one-way delays as* $\tau_1 = \Theta_1 + \gamma$ *and* $\tau_2 = \Theta_2 - \gamma$*, respectively.*

PROOF. If $i \geq 1$ or $j \geq 1$, there is no additional information to help us assign the time $(i + j) \cdot p$ to τ_1 and τ_2. Thus, if and only if $i = 0$ and $j = 0$ (i.e., $\text{RTT} = \Theta_1 + \Theta_2$), GTP can unambiguously estimate τ_1 and τ_2 as stated. □

Remark 1. The GTP programs in Algorithm 1 and 2 can run in kernel space to improve the accuracy of the timestamps $\{t_1, t_2, t_3, t_4\}$, because kernel processing typically has highest CPU priority in an OS, making it less prone to delay by high priority computation. Moreover, as Θ_1 and Θ_2 are obtained through voltage signal processing, their accuracy is subject to the resolution of the voltage signal capture. Thus, in practice, the RTT may not exactly equal $\Theta_1 + \Theta_2$. In Line 25 of Algorithm 1, the slave checks the condition in Proposition 1 by comparing $\text{RTT} - \Theta_1 - \Theta_2$ with a threshold η. From Eq. (5), ideally $\text{RTT} - \Theta_1 - \Theta_2$ is a multiple of p. Thus, we may set η to be $p/2$. If the check in Line 25 is true, the slave computes two offsets between the slave's and master's clocks as $\Delta_1 = (t_2 - \tau_1) - t_1$ and $\Delta_2 = (t_3 + \tau_2) - t_4$. Then, the slave uses the average offset $(\Delta_1 + \Delta_2)/2$ to calibrate its own clock.


(a) Θ_1 and Θ_2 in GTP.

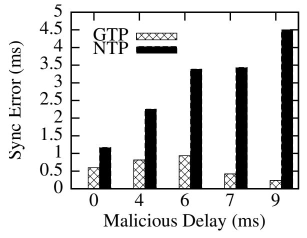

(b) Synchronization errors.

Figure 9: GTP vs. NTP in LAN.

The sufficient and necessary condition in Proposition 1 is closely related to the voltage measurements. The following proposition gives a necessary condition that depends on the RTT only. The proof can be found in Appendix B. This necessary condition allows us to quickly assess whether a cyber network can support GTP without any need to deploy the GTP hardware.

PROPOSITION 2. RTT $< 2p$ *is a necessary condition for GTP to unambiguously estimate the two one-way delays as* $\tau_1 = \Theta_1 + \gamma$ *and* $\tau_2 = \Theta_2 - \gamma$, *respectively.*

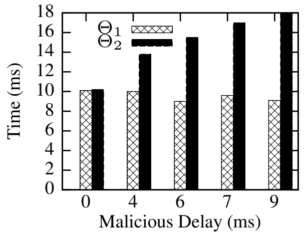
For a 50 Hz power grid, the necessary condition given by Proposition 2 is RTT < 40 ms. Section 6 presents extensive RTT measurements in our city in the absence of delay attack. The results show that the RTT rarely exceeds 40 ms. This underlines the GTP's practicality in city-scale networks. In the rest of this paper, by *GTP conditions*, we refer to the conditions given by Propositions 1 and 2.

Remark 2. From the proof of Proposition 2, although we cannot unambiguously estimate τ_1 and τ_2 in Case 2.2, Case 2.3, and Case 3, we still have useful information about τ_1 and τ_2. For instance, in Case 2.2 and 2.3, there are two possible solutions for τ_1 and τ_2: $\tau_1 = \Theta_1 + \gamma$, $\tau_2 = \Theta_2 + p - \gamma$; or $\tau_1 = \Theta_1 + p + \gamma$ and $\tau_2 = \Theta_1 - \gamma$. Thus, it is possible to assess the symmetric link assumption up to some uncertainty.

Remark 3. The synchronization error of GTP depends on the accuracy of the timestamps $\{t_1, t_2, t_3, t_4\}$, the calibration of γ, and the sound card's resolution in sampling the voltage signal. First, although hardware-level packet send/receive timestamping by the network interface card (NIC), as prescribed in the Precision Time Protocol [4], will achieve best accuracy, the requirement for special NIC hardware could hinder adoption significantly. Thus, GTP timestamps in software. Commodity PCs generally have $\pm 15\,\mu$s packet timestamp errors [5], which is quite acceptable. Second, from our WAN-scale measurements in Section 4.2, the standard deviation of γ, which characterizes its uncertainty, is about $60\,\mu$s. Third, as the sound card adopts a sampling rate of 44 kHz, the time resolution is $\frac{1}{44\,\text{KHz}} = 22\,\mu$s. Thus, we expect that GTP will achieve sub-ms (down to 0.1 ms) synchronization accuracy. This accuracy will be benchmarked in Section 5.3.

5.3 GTP Performance

We conduct comparative experiments based on the setup in Fig. 1(b) to measure the synchronization errors of GTP and NTP in the presence of asymmetric links. Note that in the experiments, both the slave and the stratum-2 time master are equipped with GTP hardware as shown in Fig. 7.


(a) Θ_1 and Θ_2 in GTP.

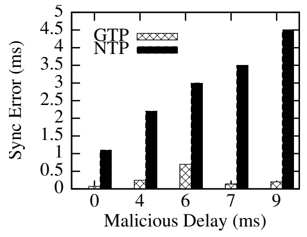
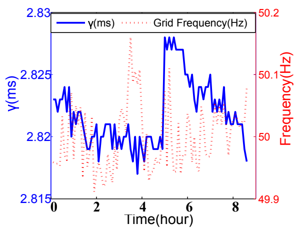
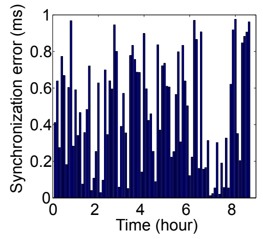

(b) Synchronization errors.

Figure 10: GTP vs. NTP in WAN.



(a) Grid frequency and $\gamma(t)$.



(b) Synchronization error.

Figure 11: Grid frequency, $\gamma(t)$, and GTP synchronization error between locations A and B, in the absence of delay attack.

We conduct two sets of experiments under the LAN and WAN settings, respectively.

In the LAN-scale experiments, all the nodes shown in Fig. 1(b) are located on the same floor of a building. We use ARP spoofing (see Section 3.2) to implement the asymmetric delay attack. The malicious delay increases from zero to 9 ms. As measured in Section 4.2.1, the angle difference γ is near-zero under this setting. Thus, we set $\bar{\gamma} = 0$ for the calculation of the slave clock offset. Fig. 9(a) shows the Θ_1 and Θ_2 measured by GTP when the malicious delay increases from zero to 9 ms. As the attack delays the reply packet, from Fig. 9(a) we can see that Θ_2 increases with the malicious delay. Fig. 9(b) shows the synchronization errors of GTP and NTP. We can see that GTP's synchronization errors are within one ms, which is consistent with our analysis in Remark 3. Note that we currently implement GTP in user space. A kernel space implementation will reduce the randomness of packet timestamping and thus GTP's synchronization errors. In contrast, as NTP is unaware of the link asymmetry, its synchronization error is about half of the malicious delay, consistent with the analysis in Section 3.1.

In the WAN-scale experiments, the slave and master nodes are at locations A and B shown in Fig. 4. From the angle difference measurements in Section 4.2.2, we set $\bar{\gamma} = 2.8$ ms for computing the slave clock offset. In the first set of WAN-scale experiments, there are no delay attacks. Fig. 11 shows the grid frequency, the $\gamma(t)$ between the two locations, and the GTP synchronization error. We note that the fluctuations of grid frequency and the γ are caused by load fluctuations and varying load distribution. From the figure, during the experiment, the γ is within $(2.81, 2.83)$ ms. Its impact on GTP's synchronization error is largely reduced by the calibration $\bar{\gamma} = 2.8$ ms. Therefore, as shown in Fig. 11(b), GTP maintains sub-ms errors. We note that several other factors discussed in Remark 3 also affect GTP's errors.

In the second set of WAN-scale experiments, we launch

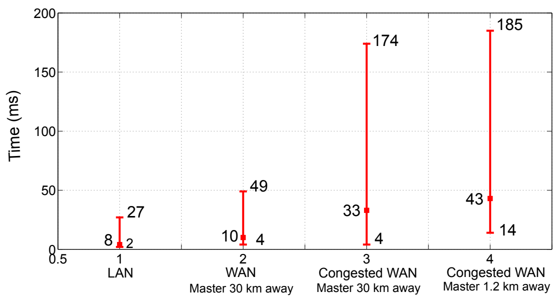


Figure 13: RTT statistics in a LAN and a city-scale WAN. Square dot represents average; error bar marks the minimum and maximum.

delay attacks. The ARP spoofing attack is launched in the LAN of the slave node. Fig. 10(a) shows the Θ_1 and Θ_2 measured by GTP when the malicious delay increases from zero to 9 ms. The Θ_2 increases with the delay. Fig. 10(b) shows the synchronization errors of GTP and NTP. Notice that the results are similar to those under the LAN setting.

In all above experiments, the GTP conditions given in Propositions 1 and 2 are satisfied. The results show that GTP can maintain sub-ms synchronization error in the presence of link asymmetry.

5.4 Impact of Grid Phase

Section 4.1 explains that most power grids have three phases, i.e., ϕ_l has three possible values of 0, $-2\pi/3$, and $2\pi/3$. If the slave and master are on different power grid phases, an additional angle difference of $-\frac{p}{3}$ ms or $\frac{p}{3}$ ms will be present, which are ± 6.67 ms in a 50 Hz grid. This additional angle difference can be considered a part of γ. Hence, GTP and its clock offset analysis in Section 5 still hold.

We note that the phase information (i.e., R-, Y-, and B-phase) of power outlets is generally available to facility operators. Viswanathan *et al.* [23] present an approach for a slave node to identify the grid phase of its power supply using sampled ac voltage signal. Based on the phase information, a GTP slave can add -6.67 ms or 6.67 ms to the calibrated γ. When the phase information is not available, GTP may have an additional ± 6.67 ms synchronization error. As GTP on the same grid phase yields a sub-ms synchronization error (cf. Section 5.3), GTP without the grid phase information can achieve a synchronization error bound of ± 8 ms in the presence of link asymmetry. In contrast, NTP in WAN exhibits errors of up to 80 ms [24].

6. RESILIENT GTP

In this section, we first present extensive measurement results to show that the GTP conditions (Section 5.2) are satisfied in a city-scale network in diverse natural settings (including congestions), in the absence of asymmetric delay attacks. However, an attack if present could add sufficient delay deliberately to breach the conditions persistently. Therefore, we will next present a resilience policy for GTP to deal with these attacks. We will also discuss key differences between GTP and NTP in terms of their security against the asymmetric delay attack.

6.1 Validating GTP Conditions

This section presents our extensive measurements to validate the GTP conditions given in Propositions 1 and 2 in the real world.

In the first set of experiments, we measure the RTT, Θ_1, and Θ_2 under normal network conditions. The first error bar in Fig. 13 shows the range of the RTT when the GTP slave and master are in the same LAN. We can see that the RTT is always smaller than 40 ms, which ensures the necessary condition in Proposition 2. Analysis of the measured RTT, Θ_1, and Θ_2 shows that the more stringent sufficient condition in Proposition 1 is also always met. Then, we perform experiments over two days under a WAN setting when the two nodes are located at locations A and B shown in Fig. 4. The slave initiates a GTP session every 20 seconds. The upper half of Fig. 12 shows the Θ_1 and Θ_2 measured by GTP over two days. The bottom half of Fig. 12 shows the values of $\Theta_1 + \Theta_2$ and the RTT measured by GTP. During the two days, the experiment encounters only three instances in which the GTP conditions are not satisfied, which are shown as the three RTT outliers in the bottom half of Fig. 12. The second error bar in Fig. 13 shows the range and average of the RTT for the duration of the experiment.

In the second set of experiments, we intentionally create network congestion to understand its impact on GTP. Under the WAN setting, we use Ostinato [3] on two other hosts in the Ethernet of the GTP slave to generate a massive amount of background traffic to congest the network. The third error bar in Fig. 13 shows the range of the RTT. The data traffic clearly causes the RTT to increase. However, a majority of the RTT measurements remain below 40 ms; further analysis shows that the more stringent sufficient condition in Proposition 1 is also satisfied in many of the synchronization sessions. We conduct also a similar experiment for another pair of GTP slave and master that are about 1.2 km apart, where the slave's Ethernet experiences intentionally introduced congestion. We obtain similar results as shown by the fourth error bar in Fig. 13.

6.2 GTP Resilience Policy

The previous experiments show that the GTP conditions are predominantly satisfied in a real-world network, even under deliberately introduced severe network congestions. To handle more extreme situations, such as malicious packet delay attacks or extremely persistent congestions that (though highly unlikely) cannot be totally ruled out, in this section we propose a resilience policy for a GTP slave to take advantage of multiple GTP masters for resilient results. The policy consists of the following two rules.

First, in each synchronization session, the slave can synchronize with multiple *default* GTP masters, and use the master that satisfies the GTP conditions and gives the minimum RTT $-\Theta_1-\Theta_2$ to calibrate its clock. With this mechanism, if one or more default GTP masters satisfy the GTP conditions, the synchronization is successful and trustworthy. As discussed in Remark 1 (Section 5.2.2), the measurements of RTT, Θ_1, and Θ_2 are subjected to packet timestamp and voltage signal processing errors, so that RTT $-$ $\Theta_1 - \Theta_2$ can be considered a quality metric. Hence, GTP chooses the best master according to the minimum RTT $-$ $\Theta_1 - \Theta_2$ for the synchronization.

Second, for a default GTP master, in case the GTP conditions are persistently not satisfied over an entire *GTP timeout* duration, the slave will replace this default master with a new *backup* GTP master. As shown in Section 6.1, even

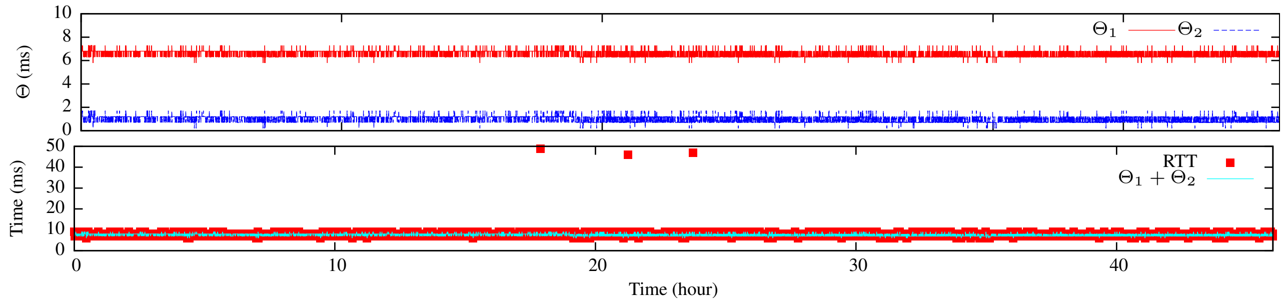


Figure 12: Θ_1, Θ_2, and RTT measured by GTP for two nodes $30\,\text{km}$ apart over two days.

under severe network congestions, we still expect the GTP conditions to be satisfied in at least some (according to the experiments, many) synchronization sessions. Hence, the majority of network congestions, particularly transient ones, will not present any problems. But should the synchronization be unsuccessful throughout the timeout duration, GTP gives an unambiguous signal to the slave to react by looking for a better master to replace the non-performing one.

We now illustrate how the second rule of the resilience policy works in practice. In this experiment, the slave resynchronizes with a default master every minute. The GTP timeout is set to be 55 minutes. Similar to the experiment in Section 6.1, we use Ostinato to create congested access to the master. In the first example, no asymmetric delay attacks are introduced. Fig. 14 shows the RTT over time. During the experiment, the GTP conditions are occasionally satisfied and the corresponding synchronization sessions are successful. Thus, the slave can keep using this default master. In the second example, we create a delay attack in which every GTP reply is maliciously delayed by 22 ms. Fig. 15 shows the RTT over time. As the GTP conditions are violated throughout the GTP timeout, the slave switches to a backup master at the end of the timeout. The new master is not under attack, and the slave knows for sure that it can successfully synchronize with the new master because the GTP conditions are met.

6.3 More Discussions on GTP vs. NTP

In this section, we analytically compare the synchronization errors of GTP and NTP. Denote by τ_1 and τ_2 the two one-way packet transmission delays. For fair comparison, we require $\text{RTT} = \tau_1 + \tau_2 < 2p$ for both GTP and NTP, to limit the impact of the link asymmetry. Under this condition, from Remark 3 (Section 5.2.2), GTP can mostly maintain a sub-ms synchronization error. This error is mainly due to uncertainty of the angle difference γ and the time granularity of voltage sampling. NTP's synchronization error is given by $\frac{\tau_1 - \tau_2}{2}$. The asymmetry $\tau_1 \neq \tau_2$ can be caused by natural route asymmetry or dissimilar load between the two network directions, or an asymmetric delay attack. In a LAN, one-way transmission delay is often small. As a result, the synchronization error $\frac{\tau_1 - \tau_2}{2}$ may reach its upper bound of p (e.g., 20 ms in a 50 Hz grid). It is significantly larger than the sub-ms error bound achieved by GTP.

7. LIMITATIONS

This section discusses the limitations of GTP. First, as GTP depends on a power grid, the nodes involved in the control of the frequency and power flows of the grid should

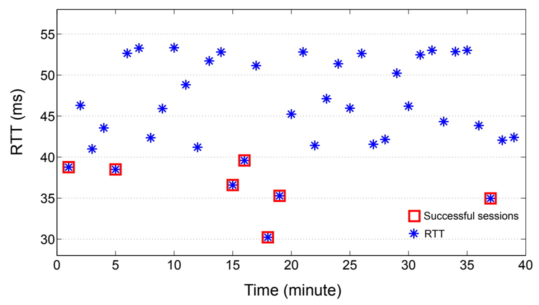


Figure 14: RTT between GTP slave and master when the master experiences network congestion. 18% synchronization sessions meet GTP conditions.

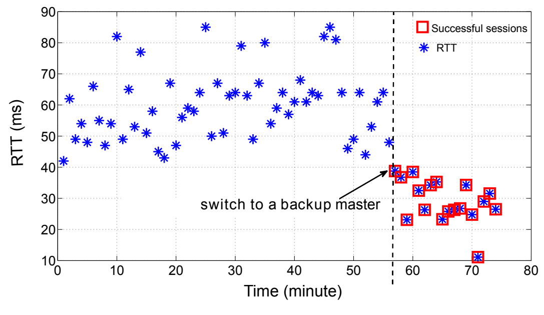


Figure 15: RTT between GTP slave and default master when the master experiences network congestion and asymmetric delay attack, respectively. In the case of the attack, the slave switches to a backup master after the timeout (55 minutes).

not employ GTP to synchronize their clocks to avoid forming a closed loop that may become unstable.

Second, the experiments in this paper are conducted in a city-scale power grid. We plan to increase the scale of GTP experiments to cover multiple cities served by the same power grid. Moreover, extension of GTP to synchronize nodes in multiple power grids that are not synchronized is an interesting subject for future work.

Third, GTP is designed to deal with the packet delay attacks under a threat model that assumes trustworthy endpoints (master and slave) including their devices for measuring the intact voltage signals. Thus, GTP is vulnerable to attacks on the voltage measurement devices, e.g., physically tampering with the circuit shown in Fig. 7 and interfering with the line-in signal using a wireless transmitter. Although addressing these attacks is out of the scope of this paper, the importance of ensuring the security of the measurement devices is noteworthy.

8. CONCLUSION AND FUTURE WORK

This paper exploited the ac power grid voltage as an extrinsic signal to tame the problem of asymmetric network delays for clock synchronization protocols based on message passing. The asymmetric delays can be caused by either natural asymmetric network routes/conditions or packet delay attacks (e.g., via ARP spoofing as demonstrated in this paper). Based on insights into the synchronization properties of the voltage signals gained from extensive measurements in a city, we designed the Grid Time Protocol (GTP) and analyzed the condition for GTP to unambiguously measure the required one-way packet transmission delays, thereby achieving resilience against asymmetric network delays.

GTP can be implemented using common ac/ac transformers and PC-grade sound cards acting as voltage sampling devices. Although inexpensive, our prototype implementation of GTP maintained sub-ms synchronization accuracy for two nodes 30 km apart in a city, when asymmetric network delays were present. We also showed that GTP admits a resilience policy that can significantly enhance its performance when multiple GTP masters are available. Specifically, if communications with any of the masters satisfy the GTP conditions, a slave's synchronization session will be successful with demonstrably trustworthy sub-ms accuracy. This desirable property is in contrast to the utility of multiple masters under NTP, whose synchronization error bound in this setting will remain as high as 20 ms. Hence, because it is inexpensive and widely applicable, GTP has promise for wide adoption in grid-connected distributed systems. In particular, it can meet the demands of applications that are currently served by NTP but desire better robustness against network dynamics or resilience against malicious attacks. Examples of these applications include a utility's metering infrastructures for end users and IoT-enabled advanced manufacturing processes.

The proof-of-concept GTP implementation presented in this paper is basic. In future work, we plan to leverage the existing code base of NTP to develop a fully deployment-ready implementation of GTP that encompasses, for example, certain NTP's built-in security features.

9. ACKNOWLEDGMENTS

The authors wish to thank our shepherd Dr. Matthias Hollick and the anonymous reviewers for providing valuable feedback on this work. This work was funded under the Energy Innovation Research Programme (EIRP, Award No. NRF2014EWTEIRP002-026), administrated by the Energy Market Authority (EMA). The EIRP is a competitive grant call initiative driven by the Energy Innovation Programme Office, and funded by the National Research Foundation (NRF).

10. REFERENCES

[1] NTP security analysis. http://bit.ly/2kQ886b.

[2] NTP: The network time protocol. http://www.ntp.org.

[3] Ostinato. http://ostinato.org/.

[4] IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2008*, pages 1–269, 2008.

[5] G. Armitage, M. Claypool, and P. Branch. *Networking and online games: understanding and engineering multiplayer Internet games*. John Wiley & Sons, 2006.

[6] D. V. Dollen. Report to nist on the smart grid interoperability standards roadmap. Technical report, Electric Power Research Institute, 2009.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI):147–163, 2002.

[8] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys*, 2003.

[9] R. Garg, A. L. Varna, and M. Wu. Seeing enf: natural time stamp for digital video via optical sensing and signal processing. In *ACM MM*, 2011.

[10] T. Hao, R. Zhou, G. Xing, M. W. Mutka, and J. Chen. Wizsync: Exploiting Wi-Fi infrastructure for clock synchronization in wireless sensor networks. *IEEE Trans. Mobile Comput.*, 13(6):1379–1392, 2014.

[11] J. Joshi. *Network Security: Know It All: Know It All.* Morgan Kaufmann, 2008.

[12] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Trans. Comput.*, 100(8):933–940, 1987.

[13] P. Kundur. *Power system stability and control*, volume 7. McGraw-hill New York, 1994.

[14] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu. Exploiting FM radio data system for adaptive clock calibration in sensor networks. In *MobiSys*, 2011.

[15] Z. Li, W. Chen, C. Li, M. Li, X.-Y. Li, and Y. Liu. Flight: Clock calibration using fluorescent lighting. In *MobiCom*, 2012.

[16] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The flooding time synchronization protocol. In *SenSys*, 2004.

[17] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Trans. Commun.*, 39(10):1482–1493, 1991.

[18] T. Mizrahi. A game theoretic analysis of delay attacks against time synchronization protocols. In *ISPCS*, 2012.

[19] A. Rowe, V. Gupta, and R. R. Rajkumar. Low-power clock synchronization using electromagnetic energy radiating from ac power lines. In *SenSys*, 2009.

[20] J.-C. Tournier and O. Goerlitz. Strategies to secure the ieee 1588 protocol in digital substation automation. In *CRITIS*, 2009.

[21] J. Tsang and K. Beznosov. A security analysis of the precise time protocol. In *Information and Communications Security*, pages 50–59. 2006.

[22] M. Ullmann and M. Vogeler. Delay attacks implication on ntp and ptp time synchronization. In *ISPCS*, 2009.

[23] S. Viswanathan, R. Tan, and D. Yau. Exploiting power grid for accurate and secure clock synchronization in industrial IoT. In *RTSS*, 2016.

[24] L. Wang, J. Fernandez, J. Burgett, R. W. Conners, and Y. Liu. An evaluation of network time protocol for clock synchronization in wide area measurements. In *Power and Energy Society General Meeting*, 2008.

APPENDIX

A. SLAVE AND MASTER PROGRAMS

Algorithm 1 Slave's pseudocode

```
// initiate a synchronization session
command sync() do
    sync_request = {}
    send sync_request to master
    $t_1$ = GetLocalTime()
end command

event sync_reply1 received from master do
    $t_4$ = GetLocalTime()
end event

event sync_reply2 received from master do
    wait until data blocks covering $t_1$ and $t_4$ are available
    locate LRZC prior to $t_1$ and compute $\Phi_1$
    locate LRZC prior to $t_4$ and compute $\Phi_4$
    $\Theta_1$ = sync_reply2.$\Phi_2$ - $\Phi_1$
    if $\Theta_1 < 0$ then
        $\Theta_1 = \Theta_1 + p$ // from the definition in Eq. (2)
    end if
    $\Theta_2$ = $\Phi_4$ - sync_reply2.$\Phi_3$
    if $\Theta_2 < 0$ then
        $\Theta_2 = \Theta_2 + p$ // from the definition in Eq. (3)
    end if
    compute RTT by RTT $= (t_4 - t_1) - (t_3 - t_2)$
    if RTT $- \Theta_1 - \Theta_2 < \eta$ then
        $\tau_1 = \Theta_1 + \gamma$, $\tau_2 = \Theta_2 - \gamma$
        $\Delta_1$ = sync_reply2.$t_2$ - $\tau_1$ - $t_1$
        $\Delta_2$ = sync_reply2.$t_3$ + $\tau_2$ - $t_4$
        clock_offset = $(\Delta_1 + \Delta_2)/2$
        use clock_offset to calibrate clock
    else
        execute the resilience policy presented in Section 6
    end if
end event
```

Algorithm 2 Master's pseudocode

```
event sync_request received from slave do
    $t_2$ = GetLocalTime()
    sync_reply1 = {}
    send sync_reply1 to slave
    $t_3$ = GetLocalTime()
    wait until data blocks covering $t_2$ and $t_3$ are available
    locate LRZC prior to $t_2$ and compute $\Phi_2$
    locate LRZC prior to $t_3$ and compute $\Phi_3$
    sync_reply2 = {$t_2$, $t_3$, $\Phi_2$, $\Phi_3$}
    send sync_reply2 to slave
end event
```

B. PROOF OF PROPOSITION 2

PROOF. We consider three cases as follows.

Case 1 ($0 \leq \text{RTT} < p$): Since $\Theta_1 \geq 0$ and $\Theta_2 \geq 0$, from Eq. (5), the non-negative integers i and j must be zero to ensure $0 \leq \text{RTT} < p$. Thus, $\text{RTT} = \Theta_1 + \Theta_2$ and GTP can unambiguously estimate the two one-way delays as $\tau_1 = \Theta_1 + \gamma$ and $\tau_2 = \Theta_2 - \gamma$, respectively.

Case 2 ($p \leq \text{RTT} < 2p$): Since $\Theta_1 \geq 0$ and $\Theta_2 \geq 0$, from Eq. (5), $i \geq 1$ and $j \geq 1$ will invalidate $\text{RTT} < 2p$. Thus, there are only three possible sub cases:

Case 2.1 ($i = 0$ & $j = 0$): Analysis same as Case 1.

Case 2.2 ($i = 1$ & $j = 0$): $\text{RTT} = \Theta_1 + \Theta_2 + p$. We cannot determine the values of i and j from all known information. Thus, GTP cannot unambiguously estimate τ_1 and τ_2.

Case 2.3 ($i = 0$ & $j = 1$): Analysis same as Case 2.2.

Case 3 ($\text{RTT} \geq 2p$): From Eq. (4), $0 \leq \Theta_1 + \Theta_2 < 2p$. Thus, from Eq. (5), $(i + j)$ must be non-zero and we cannot determine the values of i and j from the known information. Thus, GTP cannot unambiguously estimate τ_1 and τ_2.

From the above analysis, $\text{RTT} < 2p$ is a necessary condition for GTP to unambiguously estimate τ_1 and τ_2. □