Low-cost and Robust Global Time Synchronization

Marc Wyss
ETH Zurich

Marc Frei
ETH Zurich

Jonghoon Kwon
ETH Zurich

Adrian Perrig
ETH Zurich

***Abstract*—Numerous vital applications depend on accurately synchronized time, and disruptions can yield severe consequences in terms of safety and security. Yet, establishing cost-efficient and robust synchronization across geographically distributed devices is challenging. Many solutions for global time synchronization require placing trust in a single entity or system, for example in Global Navigation Satellite Systems (GNSSes) or leased infrastructure providers, constituting a single point of failure and often incurring high costs. An alternative, cost-effective solution is to run time synchronization over the Internet. However, this approach faces challenges in achieving (i) precise time synchronization, (ii) robustness to failing, misconfigured, or compromised nodes, and (iii) robustness to congestion-related issues such as volumetric DDoS attacks. Existing proposals mostly attempt to solve challenges (i) and (ii), but none provide robustness against congestion and volumetric DDoS.**

We address the challenges identified in previous work with Everdeen. Everdeen minimizes costs by running on existing Internet infrastructure and avoids relying on any single entity by enabling nodes to mutually synchronize time. The core innovation of Everdeen is its weighted neighbor-based (WNB) synchronization mode, where participants synchronize exclusively with their direct neighbors. Our evaluation shows that Everdeen provides better time synchronization quality at lower communication overhead compared to prior work. It is also considerably more robust against failing, misconfigured, or compromised hosts. Most importantly, we experimentally demonstrate that time synchronization traffic protected with Everdeen is unaffected by network congestion, including volumetric DDoS attacks.

1. Introduction

In the ever-evolving landscape of distributed systems and their diverse applications, attaining precise and reliable time synchronization is imperative for seamless coordination, data integrity, and secure communication. A wide range of applications, from ATMs to seismic event monitoring and power grid management, all depend on precisely synchronized time. Notably, in critical sectors such as telecommunications, emergency services, energy, and finance, the failure to attain synchronized time can jeopardize safety and security [1], [2], [3]. In the Russian invasion of Ukraine, for example, time synchronization is actively attacked to prevent coordination between different parts of the electric grid, resulting in outages [4].

While some services only require local synchronization, such as among servers in a data center or hosts within a single provider's network, others require synchronization across larger geographical areas, or even globally. Unfortunately, current solutions for time synchronization across large regions often necessitate placing *trust in a single entity or system*, constituting a single point of failure.

A prime illustration of this dependency is the prevalent reliance on GNSSes (Global Navigation Satellite Systems). To synchronize clocks effectively, a common approach is to deploy GNSS receivers across all servers involved in operating a distributed service. This practice essentially entrusts the reliability of timing entirely to GNSSes. Similarly, in the NTP (Network Time Protocol) architecture, the hierarchical design mainly relies on primary time servers synchronizing with GNSSes, where lower-tier servers synchronize with those higher in the hierarchy, presenting a scalable solution for achieving synchronized time even on a global scale. However, this design heavily depends on the accuracy and availability of GNSSes, which constitute the topmost layer in this hierarchical structure and thus the architecture's root of trust. As such, this design poses significant risks. Both GNSS receivers and systems are susceptible to various vulnerabilities, encompassing natural phenomena and deliberate interference. GNSS receivers are faced by threats such as jamming, spoofing, meaconing, and multipath reflections, while GNSS systems are vulnerable to malfunctions, misconfigurations, space debris, solar disturbances, and cyber attacks aimed at the satellite control infrastructure [1], [5], [6], [7], [8]. This broad spectrum of potential vulnerabilities has resulted in increased regulatory demands for GNSS independence [9], [10], [11].

To reduce this dependency, alternative solutions aim to synchronize clocks over wide area networks. Still, centralization of trust and a large variety of possible attacks pose significant challenges also to those solutions. For instance, servers running distributed services requiring synchronized time might be interconnected through leased infrastructure to counter volumetric Distributed Denial of Service (DDoS) attacks and to ensure necessary communication qualities such as symmetric latency and low jitter. While effective, leased infrastructure demands substantial trust in the infrastructure provider and incurs high costs. A more economical alternative involves clock synchronization over the public Internet. Yet, this avenue introduces potential communica-

tion disruptions due to unpredictable jitter and necessitates trust in the Internet's resilience against attacks. Such trust is often unwarranted, as synchronization methods reliant on the Internet can be susceptible to manipulation by on-path attackers who may tamper with, replay, delay, or drop packets [12], [13], [14]. Moreover, congestion can lead to asymmetric latencies and therefore erroneous time offset calculations, or even complete communication disruptions. Also, misconfigured or compromised time servers can potentially propagate inaccurate time information.

Apparently, achieving affordable global time synchronization without placing trust in a single entity or system is challenging. As we elaborate in Section 8, existing solutions from both industry and academia have proven unsatisfactory in addressing this problem. This calls for mechanisms that work in a federated way to overcome centralization and that leverage existing Internet infrastructure to allow operating at low costs. Such federated mechanisms synchronizing time over the Internet need not only be scalable, but also be able to achieve the necessary synchronization precision and be robust to faulty, misconfigured, and compromised nodes as well as to congestion, which can arise either naturally from high demand or be caused by volumetric DDoS attacks.

As a concrete instantiation of such a mechanism, we propose Everdeen.[1] Everdeen is designed to deliver synchronized time across large regions or even globally, encompassing many Autonomous Systems (ASes), i.e., independent networks forming the Internet. Compared to prior solutions, Everdeen improves synchronization performance by reducing clock skew while simultaneously lowering communication overhead. It is also more robust against a larger fraction of faulty, misconfigured, and compromised nodes in the network. Most importantly, Everdeen addresses the previously unsolved problem of protecting time synchronization traffic from congestion, a common limitation of earlier systems. Our key insight is that synchronization traffic on long inter-domain forwarding paths is much harder to protect than communication between neighboring ASes. Intuitively, the longer the path, the higher the probability of encountering at least one faulty or malicious on-path entity. Even if all on-path entities are benign, protecting communication on long paths against volumetric DDoS attacks (launched by off-path attackers) is challenging.

This insight leads us to propose that nodes only synchronize time with their immediate neighbors, rather than with all nodes. In Everdeen, a node's clock therefore influences remote nodes only through transitive propagation over multiple synchronization rounds. We refer to this as *weighted neighbor-based (WNB)* mode. In WNB, an AS does not treat synchronization results of each neighboring AS the same when deciding on how to adjust its clock. Instead, it assigns a weight to each neighboring AS that specifies the importance of the corresponding measurement results. The weights are chosen in a way that ensures provable security properties compared to prior systems. Neighbor-based synchronization allows early filtering of faulty or compromised measurements instead of propagating this information through the whole network. Furthermore, it enables effective protection against congestion: With Everdeen's *DDoS-resistant One-hop Synchronization Channel (DOSC)*, even volumetric DDoS attacks cannot cause time synchronization packets between two ASes to be delayed or dropped.

We make the following contributions:

- **Everdeen**, with its key components WNB and DOSC, the first low-cost solution for global time synchronization that is robust against faulty, misconfigured, compromised, and unavailable nodes and GNSSes, and, in particular, against congestion.
- **Formal proof** that for *every* network topology and adversary distribution, Everdeen is always *at least* as robust to faulty, misconfigured, compromised, unavailable, and actively malicious nodes as prior solutions.
- **Simulations** on the CAIDA Internet topology showing that Everdeen is $\sim 3\times$ less affected by faulty, misconfigured, or compromised time servers than existing solutions. It also achieves better skew, reduces message overhead by $\sim 2\times$, and cuts communication overhead by $\sim 4\times$, all while running on commodity hardware.
- **Evaluation** of the impact of volumetric DDoS showing that, without DOSC, time synchronization traffic is highly vulnerable; however, with DOSC, it remains working correctly even under heavy congestion.
- **Incremental deployability**, benefiting early adopters with better synchronization, performance, and security.

All our code is open-source, reproducible, and designed for simple installation, execution, and customization.[2]

2. Internet Time Synchronization Challenges

Implementing time synchronization over the Internet faces three main challenges: (i) precise synchronization, (ii) robustness against faulty nodes, and (iii) defense against congestion and DDoS attacks. This section explores how different proposed systems address these challenges and examines the limitations of each approach. An overview is given in Table 1, with further discussion in Section 8.

Precision (P). The goal of time synchronization is to ensure that distributed participants agree on a common time. As clocks naturally drift, synchronization systems must regularly exchange information to maintain consistent time across all participants. A key performance metric for these systems is clock skew, which is the maximum time difference between any two participants; lower clock skew is preferable, indicating better synchronization. Internet time synchronization faces challenges in terms of path latency asymmetry and jitter. Asymmetric paths have unequal travel times for messages in each direction, leading to incorrect time offset calculations. Jitter introduces variability in packet delay, causing fluctuations in these calculations. Internet

1. Inspired by the character Katniss Everdeen from the "The Hunger Games" series, known for her resilience and agility and for becoming a symbol of resistance in a dystopian world.

2. We will make our source code publicly available.

time synchronization is well-established, with the NTP being the most prominent example.

Fault Tolerance (F). Time synchronization needs to be robust to faulty, misconfigured, and compromised nodes. To achieve such robustness, synchronization algorithms are often designed to be Byzantine fault-tolerant [16]. In a topology corresponding to a complete graph, fault-tolerance allows those algorithms to withstand up to one-third of faulty, misconfigured, or compromised nodes, ensuring that well-functioning nodes maintain closely synchronized clocks [17]. However, the Internet does not correspond to a fully connected graph: Not all ASes are peering directly with each other, some ASes can only reach each other over longer inter-domain paths. Consequently, there might exist malicious entities on the path between two benign ASes that can influence the corresponding time synchronization measurements, for example by introducing asymmetric delay or dropping packets. The existence of longer communication paths in the Internet with potentially malicious on-path entities implies that time synchronization algorithms can only be robust against less than one third of malicious nodes.

In most of the proposed fault-tolerant time synchronization systems, any node measures its time offset with any other node, applies a fault-tolerant midpoint function, and adapts the time value of its local clock accordingly. We refer to this mode of operation as *any-to-any (A2A)* mode, which encompasses various systems, including the most pertinent and contemporary systems in the domain, for example G-SINC [18]. In this paper, we will therefore extensively evaluate Everdeen's weighted neighbor-based (WNB) mode against A2A, and thus implicitly assess Everdeen against all systems adopting the A2A mode.

Congestion and Volumetric DDoS (C). Congestion can happen naturally due to high demand or be caused by network-targeting volumetric DDoS attacks aimed at network links or routers. Due to packet drops or delayed packets, which introduce asymmetric latency, congestion can significantly disrupt time synchronization availability and precision. Such disruptions thus allow malicious entities to manipulate time offset measurements between synchronization peers. With DDoS attacks, disruptions are possible even if the entity is neither a synchronization peer nor on their communication path. We stress that DDoS attacks cannot simply be classified and handled as byzantine faults, as a single malicious entity can potentially cause disruptions between any two benign nodes in the network. While precisely targeted attacks may pose practical challenges, e.g., due to topological constraints limiting the reach of malicious entities, defending against volumetric DDoS attacks remains a complex task. Achieving availability is often considered more challenging than other security properties such as confidentiality or integrity [19].

Much work has focused on abusing time synchronization systems for DDoS attacks, such as NTP amplification attacks [20], [21], however, *little attention has been given to defending time synchronization communication against volumetric DDoS*. As discussed in the related works section, existing volumetric DDoS defense solutions have significant limitations. These include circular dependencies, where traffic can only be protected if time is already closely synchronized. Other approaches are reactive, attempting to distinguish benign from DDoS traffic to block DDoS traffic, and many are vulnerable to source address spoofing or fail to defend against legitimate-looking attack traffic. To the best of our knowledge, Everdeen is the first global Internet time synchronization system designed to fundamentally address the problem of volumetric DDoS.

TABLE 1: Various systems and the challenges they address (P: precision, F: fault tolerance, C: congestion).

Mode	System	Challenges P	F	C
other	NTP stratum model	✔	✗[a]	✗
	GTSP	✔	✗	✗
A2A	Lynch-Welch	✔	✔[b]	✗
	G-SINC	✔	✔[b]	✗
WNB	Everdeen	✔	✔	✔

[a] Clients can query multiple servers, but the NTP model remains centralized and thus vulnerable due to heavy reliance on GNSSes.
[b] Everdeen is robust against more faulty, misconfigured, compromised, or unavailable participants than A2A systems (Section 5).

3. Assumptions and Objectives

3.1. Time Synchronization Terminology

Our model is a graph $G = (V, E)$, where V is the set of nodes representing ASes, and E the set of (bidirectional) inter-domain links. We use the terms AS and domain interchangeably. Every node $v \in V$ runs a time server with a GNSS receiver and a hardware clock, where we model the latter as an increasing function $H_v(t)$ returning the clock's output at real time t. Node v has no access to t; it can only measure the progress of time by querying $H_v(t)$. We assume that the hardware clock drift is bounded, i.e., that there exists some γ such that $(1-\gamma) \leq \frac{\mathrm{d}H_v}{\mathrm{d}t}(t) \leq (1+\gamma)$. The goal of clock synchronization is to compute a logical clock L_v for every node v, such that (i) all nodes closely track the real time (accuracy), (ii) their mutual time difference is minimized (skew), and (iii) their clock rates are bounded to prevent unpredictable jumps, rapid increases, or reversals in time values. Consequently, our aim is to minimize (i) $\mathcal{A}(t) := \max_{v \in V}\{|L_v(t) - t|\}$, (ii) $\mathcal{S}(t) := \max_{v,w \in V}\{L_v(t) - L_w(t)\}$, and (iii) μ, with $(1-\mu)\frac{\mathrm{d}H_v}{\mathrm{d}t}(t) \leq \frac{\mathrm{d}L_v}{\mathrm{d}t}(t) \leq (1+\mu)\frac{\mathrm{d}H_v}{\mathrm{d}t}(t)$. This also serves the purpose of preventing trivial solutions, such as all logical clocks simply reporting a constant time.

3.2. Threat and Network Model

Threat Model. For synchronization with GNSSes, we do not make assumptions about attackers' capabilities. Attackers may use any mechanism including jamming or spoofing

to provide an arbitrary GNSS time to an AS' time server or to prevent it from receiving any time reference altogether. For network-based synchronization, we model the Internet as a graph of edges and nodes, where nodes can be benign or malicious, with every node m in the set of malicious nodes $M \subseteq V$ capable of arbitrary behavior (Dolev-Yao model).[3] Our proofs and simulations operate at this level of abstraction. In practice, nodes represent ASes, and malicious ASes (including end hosts, internal and border routers, and infrastructure services) can therefore also behave arbitrarily, i.e., eavesdrop, inject, drop, modify, or delay packets. Similarly, attackers on an inter-domain link between two ASes can manipulate traffic in any way. In such cases, our model considers the link as malicious, and implicitly also the measurement results between the respective time servers. Our model also allows any end host, including those within benign ASes or ASes not participating in Everdeen, to launch volumetric DDoS attacks against any link or router on the Internet. Ensuring the robustness of network-based time synchronization against such DDoS attacks is the primary contribution of our work.

Network Model. ASes participating in Everdeen are expected to deploy time servers and be correctly configured. Our network model assumes that neighboring ASes are directly connected through dedicated links. If AS connectivity is established by means of an Internet Exchange Point (IXP), we expect the IXP to implement mechanisms for traffic isolation, such that inter-AS communication cannot be disrupted by excessive traffic volumes of an other customer AS of the IXP. As an alternative, an IXP may instead explicitly participate in inter-domain routing by registering as an AS. Furthermore, we assume that border routers can process packets at line rate, i.e., that ingress packets do not randomly get dropped due to the router not being able to process them in a timely manner. Lastly, we assume consensus on the set of participants, ensuring that each participant observes the same set. We consider the challenge of achieving this consensus as a separate problem, orthogonal to the core focus of this paper. While for smaller topologies a shared configuration file (e.g., as provided by a commonly agreed upon party) might be sufficient, larger deployments might require a more scalable solution like dissemination of authenticated topology information among parties with limited mutual trust using smart contracts running on a blockchain [22]. Participants do not need full topology knowledge, just two key pieces: (i) the set of participants and (ii) the next hop to each destination participant.

3.3. Objectives

We aim to achieve the following objectives:

- **O1** Achieve high accuracy and low skew when GNSSes are available, i.e., minimize $\mathcal{A}(t)$ and $\mathcal{S}(t)$.
- **O2** Maintain low skew even when GNSSes are unavailable, faulty, or suffer from malicious interference.

3. For simplicity, we use 'malicious' to refer collectively to malicious, misconfigured, compromised, or faulty nodes.

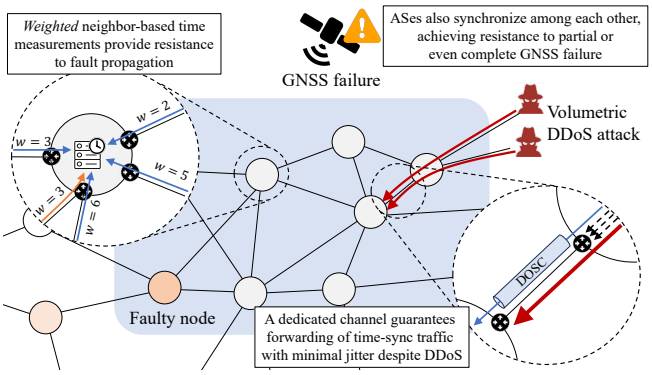


Figure 1: Everdeen overview. Nodes (ASes) keep synchronized with low skew even when GNSSes are compromised or unavailable. Everdeen also provides Byzantine fault-tolerance and robustness against congestion, including DDoS.

- **O3** Establish robustness against misconfigured, unavailable, or compromised time servers in the network.
- **O4** Protect time synchronization availability and precision against prevalent threats on the Internet, in particular against congestion caused by volumetric DDoS attacks.
- **O5** Avoid relying on single entities and centralized systems to avoid single points of failure.
- **O6** Minimize implementation and deployment costs.

As we elaborate in the related works section, no solution we are aware of simultaneously achieves all these objectives.

4. Everdeen

4.1. System Overview

This section describes the strategies implemented by Everdeen to fulfill the objectives outlined in Section 3.3, in particular the key components WNB and DOSC, and previews the subsequent content of this paper. A high-level overview of Everdeen is shown in Figure 1 and details on its algorithms can be found in Appendix A.

1. Robustness against GNSS Issues. In Everdeen, time synchronization nodes correspond to ASes. To closely align the ASes' clocks with Coordinated Universal Time (UTC), ASes in Everdeen rely on GNSSes as a time reference (O1). However, GNSSes may not always be available or accurate. Solely relying on GNSSes as a single point of trust could lead to de-synchronization of ASes during GNSS issues, resulting in low accuracy and high skew.[4] Importantly, our goal is not to enhance the robustness of GNSSes themselves, but rather to achieve globally synchronized time even in the event of GNSS failures, misconfigurations, and attacks. To reduce dependence on the GNSS ecosystem, ASes in Everdeen additionally synchronize time among themselves

4. The GNSS ecosystem is considered a single point of trust even with multiple systems, e.g., GPS, Galileo, GLONASS, and BeiDou, as jamming or disturbances like solar superstorms can impact multiple systems [8].

over the Internet. Each AS *prioritizes this network-based synchronization* over GNSS-based synchronization by allowing it to apply more substantial time corrections to its logical clock. This process is fully automatic and requires no operator intervention. A localized GNSS attack on a few ASes does not affect accuracy or skew if enough ASes continue to propagate correct time. In a global GNSS attack, accuracy degrades linearly with the duration of the outage, but *low skew among all benign nodes is always maintained* (O2).

2. Byzantine Fault Tolerance. To achieve robustness against malicious nodes, Everdeen follows a similar approach as previous Internet time synchronization solutions, and lets nodes apply a Byzantine fault-tolerant midpoint function (BFTMF) to their network-based time offset measurements (O3). In contrast to an averaging function, for example, the BFTMF allows to compute a result (the midpoint) that is unaffected by a fraction of anomalous measurements. In each synchronization round, a node corrects its logical clock according to the computed midpoint. We stress that Everdeen is independent of the specific BFTMF used. This independence also extends to any proofs and simulations presented in this work, which are abstracted from any particular BFTMF. BFTMFs can typically tolerate up to one-third of malicious inputs; as long as this threshold is not exceeded, the computed midpoint remains unaffected, regardless of the number of synchronization rounds.

3. Neighbor-based Synchronization. Everdeen introduces a new synchronization mode, the *weighted neighbor-based (WNB)* mode. In WNB, each node synchronizes time exclusively with its immediate neighbors. Thus, a node's clock value can only influence the entire network through transitive propagation over multiple synchronization rounds. Furthermore, an AS implementing WNB assigns a weight (a natural number) to each neighboring AS. The weight specifies a neighbor's importance with respect to the corresponding offset measurement, and thus influences by how much a node adjusts its clock. The weights are chosen in a way that ensures provable security properties compared to the family of all systems implementing the A2A mode (Section 2), in which nodes synchronize with all other nodes in the network. In Section 4.2 we specify WNB and show how it can be incrementally deployed. In Section 5 we prove that, with our particular choice of WNB weights, malicious information does not propagate further in the network compared to A2A. Furthermore, for the CAIDA Internet core topology, we show that WNB is up to $\sim 3\times$ as robust to misconfigured, unavailable, or compromised time servers. At the same time, WNB achieves superior clock skew while inducing $\sim 2\times$ lower message and $\sim 4\times$ lower communication overhead than the A2A mode. Lastly, Appendix B shows that WNB enables orders of magnitude faster midpoint computations than A2A.

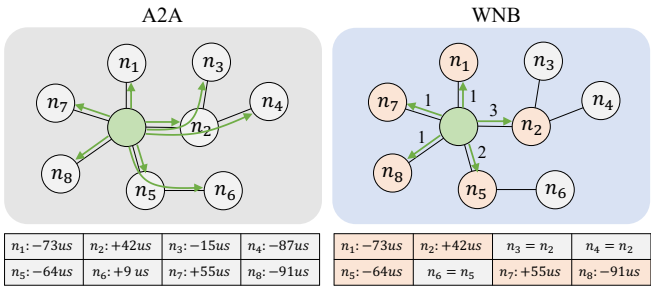
4. Robustness against Congestion. Time synchronization traffic is much more challenging to protect against congestion on long inter-domain forwarding paths compared to short ones (Section 8). In WNB, nodes synchronize time exclusively with their immediate neighbors, meaning traffic is forwarded only over one hop. This significantly simplifies protection against congestion, including volumetric DDoS attacks. This insight forms the basis of our proposed solution for defending against congestion: the *DDoS-resistant One-hop Synchronization Channel (DOSC)*. DOSC guarantees low-jitter forwarding of synchronization traffic up to a configurable bandwidth threshold (O4). Thus, volumetric DDoS cannot negatively impact accuracy or skew—*Everdeen is the first solution to achieve this in the public Internet*. In many deployments, DOSC only requires changes in router configurations, and does not require additional hardware or replacing existing one. Section 4.3 provides a detailed description of DOSC and its incremental deployment. Section 6 experimentally demonstrates that, with DOSC, time synchronization traffic and thus time offset computations are robust against DDoS.

5. Low-Cost Infrastructure. Everdeen leverages public Internet infrastructure instead of leased communication services, creating a cost-effective architecture that enables rapid expansion of the synchronization topology (O5 and O6). Notably, Everdeen operates without a hierarchy, eliminating the need to trust any single entity.

6. Deployment. We consider two primary scenarios for the deployment of Everdeen. First, deployments comprising *large geographical areas*, where Everdeen is deployed across various independent networks operated by different entities, possibly spanning different countries. The objective is to synchronize time among these networks without fully relying on GNSSes or placing trust in any single participant or third party. This deployment ensures robust synchronization while maintaining a decentralized approach, vital for scenarios where centralized trust is not acceptable. Second, deployments aiming to achieve *global time synchronization*, where Everdeen is deployed at the largest, most interconnected (core) ASes in the Internet. In this scenario, non-core ASes synchronize their time with core ASes, and end hosts obtain their time references from their respective local ASes, thus achieving the necessary scalability. We thus operate in the same setting as related work [18].

4.2. The WNB Synchronization Mode

Description. In the weighted neighbor-based (WNB) mode, each node synchronizes time exclusively with its immediate neighbors. Despite being neighbor-based, these local actions still achieve collective time synchronization. Not all neighboring ASes are equally relevant, so WNB allows to weight their measurements accordingly. In every synchronization round, each node measures the time offset to each neighbor, copies the results according to the corresponding weight, computes the midpoint of all results, and adjusts its clock towards that midpoint. This is fundamentally different from A2A, where all nodes synchronize time with one another. Nodes in WNB therefore transmit much fewer messages per synchronization round, and as a consequence, they can afford to synchronize more frequently to their neighbors while at the same achieving better skew and lower message- and communication overhead.



n_1: $-73us$	n_2: $+42us$	n_3: $-15us$	n_4: $-87us$
n_5: $-64us$	n_6: $+9\ us$	n_7: $+55us$	n_8: $-91us$

n_1: $-73us$	n_2: $+42us$	$n_3 = n_2$	$n_4 = n_2$
n_5: $-64us$	$n_6 = n_5$	n_7: $+55us$	n_8: $-91us$

Figure 2: Comparison between A2A and WNB. With A2A, the green node measures time offsets to all other nodes and stores them in an array. With WNB, it only measures the time offset to each direct neighbor, where it stores the result in all array slots that correspond to nodes for which the communication path in A2A traverses that neighbor.

Weight Selection. The most important parameters in WNB are the weights, which raises the question of how each node should choose them, as there are many possible strategies. The weights could be chosen to optimize a specific metric, such as simplicity, low skew, or faster synchronization convergence, for a specific family of topologies. However, this would complicate a general comparison between WNB and previous work, such as A2A. We require a weight selection strategy that allows to formally prove WNB's superiority over A2A *in any possible topology*. Our primary metric for comparison is robustness to faulty, misconfigured, and compromised participants (Section 2). For this metric, the weight selection strategy should enable a formal comparison between WNB and A2A, *regardless of the adversary distribution*. We found that WNB achieves provable properties when each node assigns a measurement result from a specific neighbor n_i a weight of w_i, where w_i refers to the number of paths that would traverse the neighbor during synchronization with all ASes in A2A. This A2A-dependent WNB weight assignment is necessary to derive a generally applicable statement. After copying the results according to their weights, the number of entries in the array for the midpoint function is the same as in A2A. An example of how WNB weights are derived is shown in Figure 2.

Incremental Deployment. History shows that the deployment of new Internet technologies can be challenging without an incremental approach. We offer a concrete deployment path, where an AS can follow this path independently of other ASes: (i) An AS starts by synchronizing only to GNSSes. (ii) It then deploys a time server and synchronizes with other, potentially non-adjacent, ASes that have a time server deployed via A2A. (iii) *The AS synchronizes with participating neighboring ASes via WNB and with other ASes via A2A*, i.e., ASes to which the respective forwarding paths lead through non-participating neighbors. Thereby, it stores the measured offset to a neighbor n_i as many as w_i times, while it stores the offsets measured to a remote AS, for which the forwarding path leads through a neighbor without time server deployed, only once. The AS does not synchronize with remote ASes to which the forwarding path would lead through a participating neighbor. (iv) Finally, if all neighbors have a time server deployed, the AS adopts WNB for synchronization with all of them. Thereby, steps (i) and (ii) minimize entry barriers, while steps (iii) and (iv) incentivize WNB adoption by gradually improving efficiency and security. Specifically, ASes that switch from A2A to WNB, even if only for a single link to some neighbor, gain robustness against malicious nodes, as the reasoning behind our proofs in Section 5.1 also applies to incremental deployment; reduce communication overhead since synchronization over this link is only needed to the neighbor; improve skew and accuracy due to the ability to synchronize more frequently (a consequence of the lower communication overhead); and enhance robustness against DDoS attacks through the use of DOSC.

Synchronization Precision. For network-based time synchronization, the achieved synchronization precision depends on the jitter introduced by (i) the time server and (ii) the network. However, only the jitter introduced by the network poses problems in practice. The time server is fully under the control of its operator, and ingress- and egress timestamping can remove potential timing variations. Therefore, if a neighboring server has high jitter, this does not imply that a node achieves lower precision with WNB than with A2A. Network elements such as routers, on the other hand, can receive different traffic volumes depending on remote traffic sources' demand or also due to volumetric DDoS attacks. This can lead to queues building up, causing asymmetric latency, or packets to get dropped completely. If available, existing solutions such as PTP can mitigate the effect of queuing delay during mild congestion. During volumetric DDoS attacks, however, legitimate time synchronization packets are not necessarily delayed, but get dropped in the first place. To also achieve forwarding guarantees during volumetric DDoS, DOSC is required. We highlight that network communication in A2A experiences the same or higher jitter to the next hop as WNB (if DOSC is deployed then jitter is minimal in WNB), plus additional jitter due to the longer forwarding paths.

Limitations. Transitioning from A2A to WNB comes at the cost of a higher impact of time server failures. In A2A, the failure of a neighboring AS' time server has minimal impact, given the numerous other ASes available for time synchronization. However, in WNB, the complete loss of a neighboring time server means that w_i measurements are unavailable. The actual impact may not be severe due to ASes often being highly interconnected, which ensures the availability of multiple neighbors with functioning time servers. While deploying backup servers is a practical solution, the simplest approach is to revert to step (iii) of the incremental deployment process, using A2A to synchronize over the affected neighbor until the neighboring AS' time server is restored. In WNB, an AS can monitor time differences with its neighbors to detect deviations. This is more manageable than monitoring hundreds or thousands of ASes in A2A. An AS can take meaningful actions when it knows its neighbors (e.g., contacting the corresponding network operator), but for a remote AS, the appropriate response may be less

clear. In cases where synchronization with both GNSSes and other ASes in the Internet is temporarily disrupted, an AS must rely solely on its local clock until connectivity is re-established. Lastly, we would like to clarify that entities surrounded solely by attackers cannot achieve synchronized time, as attackers could easily disrupt communication by dropping all packets. This scenario could apply to edge ASes with only a few (malicious) neighbors or end hosts within a malicious AS. This is not specific to Everdeen, but applies to any synchronization protocol.

4.3. Robustness against Congestion with DOSC

Overview. In this section, we show how effective mitigation of congestion, including volumetric DDoS attacks, is possible due to (i) WNB communication being strictly neighbor-based and (ii) the minimal bandwidth required for time synchronization traffic. We refer to our proposed solution as DDoS-resistant One-hop Synchronization Channel (DOSC), a highly available bidirectional channel for time synchronization traffic between neighboring ASes. Up to a certain bandwidth, traffic sent over a DOSC-protected channel is guaranteed not to be dropped and to maintain minimal latency and jitter, even in the presence of volumetric DDoS attacks. As such, it can be considered a scalable and cost-effective alternative to a direct physical cable connecting the two neighboring time servers. DOSC can be implemented through simple router configurations, without the need for additional hardware or router replacements.

Minimizing Latency and Jitter. DOSC protects communication between neighboring time servers by leveraging Differentiated Services (DiffServ) [23]. Specifically, a time server protects its traffic directed to a neighboring AS' time server by marking the IP headers of its packets with the expedited forwarding differentiated services codepoint (EF DSCP, value 46). All devices along the communication path—internal routers, border routers, and the destination time server—prioritize these packets according to the per-hop behavior (PHB) associated with their DSCP value. This ensures forwarding with minimal latency and jitter.

Authentication. DOSC takes an unconventional yet practical and effective approach to *mitigate the impact* of source address spoofing: routers either (i) clear the DSCP fields of untrusted traffic or (ii) enforce an upper bound on the rate of untrusted DSCP-enabled traffic such that it cannot cause any harm in terms of its volume. Specifically, internal routers allow only the time server to set the DSCP field, resetting it for all end hosts. Resetting is enforced at specific router interfaces, rather than based on source addresses, to prevent spoofing. For DSCP-enabled traffic originating within the domain (i.e., from a local interface), a border router can thus confidently attribute it to its AS' time server. For DSCP-enabled traffic arriving from outside the domain (i.e., from a non-local interface $\texttt{n}$), a border router does not make any assumptions regarding its authenticity but instead rate limits the traffic to a rate $\texttt{B}(\texttt{n})$, thus preventing excessive volumes. Additionally, a border router resets the DSCP fields of traffic forwarded between two non-local interfaces. DOSC supports any IP-based synchronization protocol, making source and payload authentication at time servers complementary, with existing solutions like NTS [24] or DRKey [25], [26] easily integrated. While we cannot claim to solve source address spoofing across the Internet, DOSC *mitigates its impact in the specific context of time synchronization between neighboring ASes*.

Channel Bandwidth. Border routers impose an upper limit on the rate of DSCP-enabled traffic between neighboring ASes' time servers. Specifically, for each interface $\texttt{n}$ connected to a neighboring AS, the router allocates bandwidth $\texttt{B}(\texttt{n})$ in both directions between interface $\texttt{n}$ and its internal interface for traffic marked with the DSCP field. This allocation represents the maximum allowed bandwidth, meaning any DSCP-enabled traffic exceeding this rate will be dropped. The value of $\texttt{B}(\texttt{n})$ is determined based on the time synchronization bandwidth requirements between the neighboring ASes. In typical WNB configurations (Section 5), as little as $10\,\mathrm{kbit/s}$ is sufficient to meet Everdeen's time synchronization needs.

DDoS Defense. DDoS by end hosts targeting synchronization traffic are ineffective because their traffic is always forwarded without DSCP enabled, preventing it from impacting priority queues at either routers or time servers. Similarly, DDoS attacks from neighboring ASes have no effect when DSCP is disabled. For DSCP-enabled traffic arriving from border router interface $\texttt{n}$, the neighboring AS is restricted to a very low transmission rate, $\texttt{B}(\texttt{n})$. As a result, even if hundreds of neighboring ASes send large volumes of DSCP-enabled traffic, the total traffic volume, after applying rate limits ($\sum_n \texttt{B}(\texttt{n})$), remains constrained to just a few megabits per second. *This is feasible only due to the extremely low bandwidth requirements for time synchronization traffic between neighboring ASes.* Additionally, end-to-end authentication at the time servers prevents reflection attacks, where multiple malicious ASes neighboring AS A spoof their source addresses to impersonate the time server of AS N (a benign neighbor of AS A). This would cause traffic to be reflected from AS A's time server toward AS N instead of the attacker, causing many legitimate packets to be dropped due to the bandwidth limits imposed between AS A and AS N.

Coordination and Configuration. In contrast to related work on forwarding guarantees in the Internet, DOSC only requires *coordination between neighboring ASes*. Key elements requiring coordination include the choice of synchronization protocol (e.g., NTP), the channel bandwidth $\texttt{B}$, the IP addresses of the time servers, and the shared authentication keys (e.g., for NTS). DOSC is built upon DiffServ and rate limiting, both of which are well-established and widely available components. DiffServ has been extensively deployed across numerous networks [27] and is commonly supported on modern routers [28], [29], as is rate limiting [30], [31]. This means existing routers can be reused, needing *only to be reconfigured* for use in DOSC. Internal routers may not require any modifications at all, as they typically prioritize packets marked with DSCP 46 by default, placing them in a dedicated priority queue. Updates to

router configurations are expected to be infrequent, typically occurring only when establishing Everdeen peering relationships with neighboring ASes. The ability to reconfigure existing routers stands in stark contrast to the significant costs associated with private or leased infrastructure, as well as with related work on network availability, which often has more demanding requirements—such as replacing traditional routers with programmable switches [32], implementing cryptographic support at routers [33], or managing millions of queues [34]. Given the critical role of time synchronization in the proper functioning of many applications, both critical and non-critical, we consider the configuration overhead of DOSC to be justifiable.

Contribution. We observe that the neighbor-based nature of time synchronization with its low bandwidth requirements allows for a cost-effective and robust approach to mitigating DDoS attacks by leveraging DiffServ alongside traffic policing. Unlike many existing DDoS defense mechanisms (Section 8), which are reactive—allowing attacks to continue until they are (hopefully) detected and blocked—DOSC adopts a proactive approach. By design, it prevents attacks from causing any harm, making it inherently robust against congestion. Also, DOSC does not depend itself on synchronized clocks, avoiding the circular dependencies found in prior solutions.

5. Evaluation of WNB

We first prove that, for any topology, WNB is at least as robust against faulty, misconfigured, compromised, or actively malicious participants as A2A. Then, for the particular topology consisting of the most highly connected ASes in the Internet, we evaluate the attack resistance, accuracy, skew, and communication overhead achieved by WNB and A2A, both with and without GNSS issues.

5.1. Proof of Robustness

Description. We prove that in WNB, adversely affected information does not propagate further in the network compared to A2A. The proof is applicable to any connected graph and fault-tolerant function. The proof relies on our specific selection of WNB's weights.

Measurement-related Notation. We extend the general notation introduced in Section 3.2. We define a path p from node S to node D of length ℓ as a list of nodes $[S, ..., D]$ where $\forall_{i=0}^{i<\ell-1} : (p_i, p_{i+1}) \in E$. With this notation, $p_0 = S$, $p_{\ell-1} = D$, and p_1 and $p_{\ell-2}$ are neighbors of S and D, respectively. We define $\widetilde{p}$ as the set of nodes in p without the source node: $\widetilde{p} := \{p_i\}_{1 \le i \le \ell-1}$. In A2A, a node measures the time offsets to all other nodes, potentially across different paths, storing them in an array denoted as A. An entry in A is referred to as A_p, where p defines the specific communication path to the destination node. Therefore, A is not limited to a single path per destination, but there could be multiple time synchronization measurements over multiple different paths to the same destination. The set of all paths used for measurements in A2A is denoted as P. Contrarily, in WNB, a node only measures the offset to each of its neighbors n_i. It then adds the measured offset w_i times to its array W, resulting in an array length identical to A (Figure 2). While the entries corresponding to the measurements to a neighboring AS are the same in W and A, W can contain several duplicates: $\forall p \in P : W_p = A_{[p[0],\ p[1]]}$.

Threat-related Notation. We define a path as *compromised*, if it traverses at least one malicious node. This is derived from the threat model, wherein a malicious destination can spoof time synchronization replies and malicious on-path nodes can delay synchronization packets to skew the measurements:

$$m(p) = 1 \iff \exists a \in \widetilde{p} : \ a \in M \tag{1}$$

We define an array entry as compromised if and only if the path over which it was measured is compromised:

$$m(A_p) = 1 \iff m(p) = 1 \tag{2}$$
$$m(W_p) = 1 \iff m([p[0], p[1]]) = 1 \tag{3}$$

Lastly, we assume an arbitrary but reasonable fault-tolerant function $f(P, P_M)$, that returns 1 if it can tolerate $P_M \subseteq P$ of malicious paths. Here *reasonable* means that if the function can tolerate a set of malicious paths, then it can also tolerate any subset of it:

$$\forall P_M^* \subseteq P_M : \quad f(P, P_M) = 1 \implies f(P, P_M^*) = 1 \tag{4}$$

An equivalent formulation would be that if the function cannot tolerate some set of malicious paths, then it cannot suddenly tolerate the same set plus additional malicious paths.

Proof. We first prove Theorem 1, which states that every compromised measurement entry W_p in array W of WNB implies a compromised measurement entry A_p in array A of A2A. Then, in Corollary 1, we show that any fault-tolerant function capable of tolerating the compromised entries in A can similarly tolerate the compromised entries in W.

Theorem 1. $\forall p : \ m(W_p) \implies m(A_p)$.

Proof. Consider any path $p \in P$. Then:

$$m(W_p) \iff m([p[0], p[1]]) = 1 \tag{5}$$
$$\implies \exists a \in \widetilde{p} : \ a \in M \tag{6}$$
$$\iff m(p) = 1 \tag{7}$$
$$\iff m(A_p) \tag{8}$$

Step (5) holds by definition of (3). Step (6) relies on the presence of a malicious on-path node when the first on-path node is malicious. Steps (7) and (8) follow by definition of (1) and (2), respectively. □

Corollary 1. *In WNB, adversely affected information does not propagate further in the network compared to A2A.*

Proof. Let $P_A \subseteq P$ and $P_W \subseteq P$ be the set of malicious paths in A and W, respectively. Theorem 1 implies that

$P_W \subseteq P_A$, and therefore, as f is a reasonable function according to (4), we have:

$$f(P, P_A) = 1 \implies f(P, P_W) = 1 \tag{9}$$

So if the output of a fault-tolerant function is unaffected by malicious nodes in A2A, it remains unaffected in WNB as well, preventing propagation of malicious information. □

5.2. Implementation

Description. We implement a simulator running A2A and WNB on (i) realistic inter-domain topologies consisting of either the 2000 highest-degree Tier-1 and Tier-2 ASes according to the CAIDA AS relationships data set [35] or (ii) random power-law degree graphs, which are found to well approximate real-life technological networks [36]. The latter enables testing WNB and A2A on topologies of various sizes, which can for example be used to speed up the measurements process by running them on smaller topologies. In our topologies, a core AS corresponds to a node, and an inter-domain link to an edge. All the measurement results presented in this section were performed on the CAIDA 2000 topology, where nodes always choose the shortest path towards their destination. Detailed characteristics for this topology can be found in Appendix C.

Our simulator is written in Go [37] and is highly parallelizable, thus achieving low completion times even on low-end off-the-shelf devices. Together with the simple installation and execution steps, this contributes to the reproducibility of our work. We execute all our measurements on a device with a 12th Gen Intel Core i7-1260P CPU (12 cores, 3.4 GHz) and 32 GiB of memory. We sample variables such as the network error and clock drifts at random.

5.3. Malicious Information Propagation

Description. We first assess how much adversely affected information propagates in a realistic topology. While our proofs in Section 5.1 show that WNB performs at least as well as A2A in this metric irrespective of the topology, this experiment focuses on obtaining empirical results using the CAIDA 2000 topology. We randomly select malicious nodes in the network, varying their proportion from 5% to 40%. Subsequently, we execute time synchronization processes for both A2A and WNB. We consider a benign node to become malicious if more than one third of the time measurements traverse at least one malicious node. Given the transitive impact malicious nodes can have on others, we iterate this process until a stable state is achieved. Each such measurement is repeated for a total of $R = 100$ iterations.

Results. Figure 3 shows that, for the same proportion of initially malicious nodes, WNB significantly reduces the impact on benign nodes compared to A2A. This difference is particularly evident with 10 % initially malicious nodes, where WNB affects 3.26 times fewer benign nodes than A2A. As anticipated, both protocols struggle to protect against more than one third of initially malicious nodes.

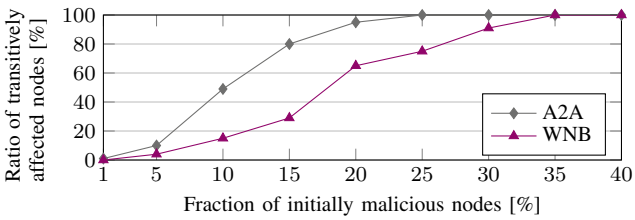


Figure 3: Ratio of transitively affected to initially unaffected nodes, given a fraction of initially malicious nodes.

5.4. Time Synchronization Performance

Description. Section 5.3 evaluated the set of (transitively) malicious nodes. For these malicious nodes, the behavior is straightforward: the initially malicious ones can act arbitrarily, while the transitively affected ones become desynchronized at a rate bounded by the sum of the maximum drift, the maximum allowed GNSS-based correction, and the maximum permitted network-based correction. Therefore, we now focus on the set of benign nodes. By definition, their time does not get affected by any of the malicious nodes. We aim to verify that they achieve desirable time synchronization quality and overhead. In the following experiments, we assess the accuracy, skew, as well as message- and communication overhead achieved by WNB in comparison with A2A. For the GNSSes, we evaluate and compare two extreme cases on the availability spectrum. In the first case, GNSSes are available and report correct results. In the second case, GNSSes are globally unavailable. Appendix D expands our evaluation by considering globally operating attackers specifically targeting GNSSes.[5]

Measurement Parameters. Initially, the simulator assigns random clock times to the nodes, resulting in an initial skew of $\mathtt{S}_0$. The simulator then performs time synchronization measurements at regular intervals of duration $\mathtt{I}$, for a total of $\mathtt{T}$ time synchronization iterations. In each interval, the simulator generates random drift values for every node's clock from a distribution within the range of [-$\mathtt{D}$, $\mathtt{D}$], and assigns network delay asymmetries to every edge from a random distribution within the range of [-$\mathtt{E}$, $\mathtt{E}$]. At the end of each interval, the simulator reads the nodes' clock values in order to calculate skew and accuracy. Subsequently, every node's clock is adjusted based on two factors: time obtained from GNSSes, with a maximum correction limit of $\mathtt{C}^{\mathtt{Sat}}$, and data collected from Internet measurements, with a maximum correction limit of $\mathtt{C}^{\mathtt{Net}}$. We emphasize that, in order to successfully counter clock drift, it is necessary that $\mathtt{C}^{\mathtt{Net}} > \mathtt{D}$ and $\mathtt{C}^{\mathtt{Sat}} > \mathtt{D}$, and to achieve resistance against faulty, misconfigured, unavailable, and potentially malicious GNSSes, it is crucial to set $\mathtt{C}^{\mathtt{Net}} > \mathtt{D} + \mathtt{C}^{\mathtt{Sat}}$. We define the absolute message overhead of A2A and WNB, respectively,

5. Because Everdeen prioritizes network-based synchronization over GNSS-based synchronization, it achieves synchronized time, i.e., bounded skew, irrespective of the values provided by the GNSSes.

TABLE 2: Description of parameters ("Para.") and their values ("Val.") for the skew, accuracy and overhead experiments.

Para.	Description	Val. A2A	Val. WNB
$\mathtt{I}$	Synchronization interval	3 s	0.6 s
$\mathtt{T}$	Nr. sync. iterations	1200	6000
$\mathtt{D}$	Max. drift per interval $\mathtt{I}$	0.83 ms	0.166 ms
$\mathtt{C}^{\mathtt{Sat}}$	Max. GNSS correction	1 ms	0.2 ms
$\mathtt{C}^{\mathtt{Net}}$	Max. network correction	2 ms	0.4 ms
$\mathtt{S}_0$	Initial skew	4 s	
$\mathtt{O}^{\mathtt{M}}_{\mathtt{REL}}$	Rel. message overhead	2	
$\mathtt{O}^{\mathtt{C}}_{\mathtt{REL}}$	Rel. comm. overhead	4	
$\mathtt{E}$	Max. per-link error	0.1 ms	
$\mathtt{X}$	Network cutoff	0.2 ms	

and their relative overhead as follows:

$$\mathtt{O}^{\mathtt{M}}_{\mathtt{A2A}} := \frac{\mathtt{m}_{\mathtt{A2A}}}{\mathtt{I}_{\mathtt{A2A}}}, \qquad \mathtt{O}^{\mathtt{M}}_{\mathtt{WNB}} := \frac{\mathtt{m}_{\mathtt{WNB}}}{\mathtt{I}_{\mathtt{WNB}}}, \qquad \mathtt{O}^{\mathtt{M}}_{\mathtt{REL}} := \frac{\mathtt{O}^{\mathtt{M}}_{\mathtt{A2A}}}{\mathtt{O}^{\mathtt{M}}_{\mathtt{WNB}}}$$

Here, $\mathtt{m}_{\mathtt{A2A}}$ and $\mathtt{m}_{\mathtt{WNB}}$ refer to the number of time synchronization messages sent by all nodes in a single iteration for A2A and WNB, respectively. With CAIDA 2000 and the shortest-path strategy, $\mathtt{m}_{\mathtt{A2A}} \approx 4 * 10^6$ and $\mathtt{m}_{\mathtt{WNB}} \approx 4 * 10^5$. Similar to the message overhead definitions, we define the absolute communication overhead of A2A and WNB, as well as their relative overhead, as follows:

$$\mathtt{O}^{\mathtt{C}}_{\mathtt{A2A}} := \frac{\mathtt{e}_{\mathtt{A2A}}}{\mathtt{I}_{\mathtt{A2A}}}, \qquad \mathtt{O}^{\mathtt{C}}_{\mathtt{WNB}} := \frac{\mathtt{e}_{\mathtt{WNB}}}{\mathtt{I}_{\mathtt{WNB}}}, \qquad \mathtt{O}^{\mathtt{C}}_{\mathtt{REL}} := \frac{\mathtt{O}^{\mathtt{C}}_{\mathtt{A2A}}}{\mathtt{O}^{\mathtt{C}}_{\mathtt{WNB}}}$$

Here, $\mathtt{e}_{\mathtt{A2A}}$ and $\mathtt{e}_{WNB}$ refer to the sum of edges traversed by all time synchronization messages sent by all the nodes in one iteration, for A2A and WNB, respectively. In our setting, $\mathtt{e}_{\mathtt{A2A}} \approx 8 * 10^6$ and $\mathtt{e}_{\mathtt{WNB}} = \mathtt{m}_{\mathtt{WNB}} \approx 4 * 10^5$. Therefore, with $\mathtt{I} = \mathtt{I}_{\mathtt{A2A}} = \mathtt{I}_{\mathtt{WNB}}$, we would have $\mathtt{O}^{\mathtt{M}}_{\mathtt{REL}} \approx 10$ and $\mathtt{O}^{\mathtt{C}}_{\mathtt{REL}} \approx 20$. To partially compensate for this, we use a synchronization interval for WNB that is five times shorter (with five times lower drift per interval), such that $\mathtt{O}^{\mathtt{M}}_{\mathtt{REL}} \approx 2$ and $\mathtt{O}^{\mathtt{C}}_{\mathtt{REL}} \approx 4$. This means that in A2A, nodes still disseminate twice as many messages, and edges still carry four times as many messages as in WNB, thus favoring A2A in our experiments. Unless otherwise specified, our experiments use the parameter configuration detailed in Table 2. We discuss the rationale behind our assumed clock drift at the end of this section. A full description of the A2A and WNB algorithms can be found in Appendix A.

Accuracy. The results of the accuracy measurements are shown in Figure 4. Both A2A and WNB achieve fast convergence despite the initial skew of 4 s and attain good accuracy when GNSSes are available.[6] Specifically, A2A achieves an accuracy of approximately 830 µs, while WNB reaches about 160 µs. The superior performance of WNB can be attributed to the fact that nodes synchronize more frequently with GNSS time, even though a smaller maximum correction is applied. If WNB nodes synchronized with GNSS sources as infrequently as in A2A and used the same maximum correction, we would expect the same accuracy as in A2A. When GNSS sources are unavailable, the achieved accuracy worsens in both modes. However,

6. The y-axis is logarithmic; convergence is actually linear.

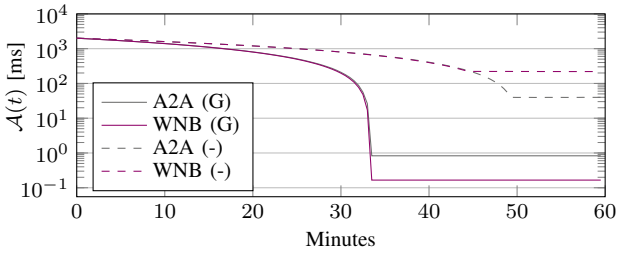


Figure 4: Accuracy $\mathcal{A}(t)$ in milliseconds for A2A and WNB, with (G) and without (-) GNSSes available.

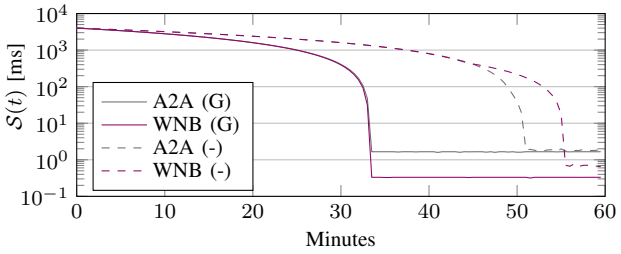


Figure 5: Skew $\mathcal{S}(t)$ in milliseconds for A2A and WNB, with (G) and without (-) GNSSes available.

this trend is more pronounced in WNB (220 ms) compared to A2A (39 ms). This is expected, as the nodes' clocks are initially set with a random distribution around the real time, and in the first round of A2A, every node performs time synchronization measurements with every other node, thus closely approximating the real time. Nevertheless, when GNSS sources are absent, maintaining accuracy is impossible due to the continuous clock drift. However, achieving low clock skew is often much more important for applications relying on globally synchronized time.

Skew. The results in Figure 5 show that A2A and WNB effectively converge and maintain low clock skew across all scenarios, with WNB notably outperforming in this critical metric. When GNSS references are available, A2A achieves a clock skew of ~1.6 ms, while WNB reaches ~300 µs. In the absence of GNSS sources, A2A and WNB stabilize at ~1.7 ms and ~700 µs, respectively. Nodes in WNB send significantly fewer messages per synchronization round compared to A2A, allowing them to synchronize more frequently with their neighbors. Interestingly, WNB achieves lower message overhead and better skew during GNSS outages than A2A. Both A2A and WNB measurements were conducted using the same per-hop network error distribution. However, in a real-world deployment, A2A would likely experience higher jitter than WNB, as DOSC prioritizes time synchronization traffic in WNB, keeping jitter minimal. Thus, the simulation setup is more favorable to A2A regarding accuracy and skew measurements.

Communication Overhead. In the previous experiments, we used $\mathtt{O}^{\mathtt{M}}_{\mathtt{REL}} = 2$ and $\mathtt{O}^{\mathtt{C}}_{\mathtt{REL}} = 4$. We now aim to explore the impact on the clock skew when varying these overheads. Specifically, we want to determine how much WNB could enhance the nodes' skew if it operated with

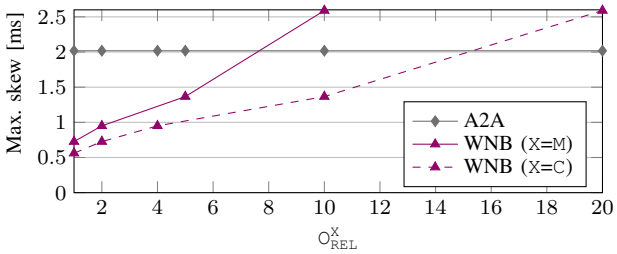


Figure 6: Max. skew in milliseconds during converged state without GNSSes available, in relation to O^{X}_{REL} ($X \in \{M, C\}$).

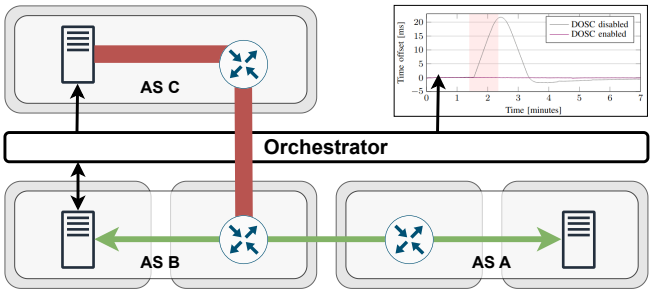


Figure 7: DOSC testbed on AWS. Grey boxes are EC2 instances.

the same overhead as A2A. Conversely, we also aim to find out how much we can reduce WNB's overhead while still achieving superior skew. To answer these questions, we vary I_{WNB} while keeping I_{A2A} unchanged. We deactivate GNSS references to isolate the impact of the different overheads on clock skew.

The results of this evaluation are presented in Figure 6. We can observe that reducing O^{M}_{REL} from 2 to 1, meaning that WNB disseminates as many messages per time interval as A2A, leads to an improvement in maximum skew by 225 µs. Conversely, increasing O^{M}_{REL} from 2 to approximately 7.6, such that WNB disseminates 7.6 times fewer messages per interval than A2A, results in a skew matching that of A2A (2 ms). Similar outcomes can be observed for the communication overhead. Specifically, reducing the O^{C}_{REL} from 4 to 1 enhances WNB's maximum skew by 410 µs. Conversely, increasing O^{C}_{REL} from 4 to approximately 15.3 leads to a maximum skew of 2 ms, which is in line with A2A's performance.

Assumed Clock Drift. In our configuration (Table 2), we assume a maximum clock drift of approximately 24 s per day. This conservative assumption serves two main purposes: demonstrating WNB's minimal hardware requirements for seamless integration into AS infrastructure and highlighting its relative advantages over A2A. Although even low-cost clocks often found in commodity servers can achieve lower clock drift, recent research has introduced software-based solutions capable of further reducing clock drift in commodity servers by a factor of around 2000 [38]. Implementing such solutions in our setup would significantly enhance WNB's clock skew performance while reducing its message and computation overhead. Moreover, absolute results in terms of skew and accuracy can be further improved by, for example, increasing the synchronization frequency.

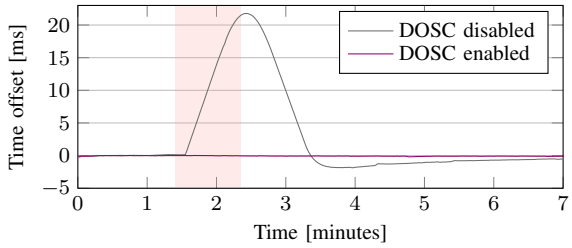
6. Evaluation of DOSC

In this section, we implement DOSC and evaluate its robustness against volumetric DDoS attacks. Our results demonstrate that, without DOSC, these attacks can shift time offsets by tens of milliseconds. In contrast, DOSC effectively mitigates such attacks, maintaining accurate time synchronization.

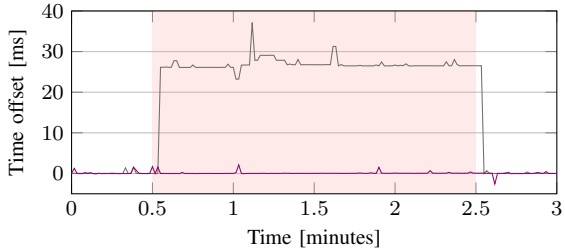
Implementation. To implement DOSC, we need to configure border routers to (i) classify packets according to their DSCP field, (ii) implement rate limiting for traffic with a DSCP value of 46, and (iii) forward packets passing this check with strict priority. While all those features are available on commercial routers [39], [40], we rely on a software-based implementation for reproducibility. Concretely, we extend an existing border router implementation written in Go with the aforementioned configuration. With the parameters from Table 2, the traffic volume per inter-domain link for WNB when using NTP is only on the order of a few kilobits per second. We therefore select a conservative value of 100 kbit/s for the DOSC bandwidth enforced by the border routers' rate limiters. We implement both a measurement client and server, where the client conducts network-based time offset measurements to the server based on NTP. Software timestamping in the kernel is used to remove side effects of the time synchronization applications on the measurements. To demonstrate that DOSC is generally applicable and not tied to one specific implementation, we also implement a version of the border router only based on the traffic control subsystem in the Linux kernel [41] and conduct the time offset measurements using the state-of-the-art NTP server chrony [42].

Testbed. We perform our evaluation on a reproducible AWS measurement testbed (Figure 7). We run three ASes, AS A, AS B, and AS C, each based on production-ready routing software deployed on a dedicated AWS EC2 instance. On another two EC2 instances, we execute the time synchronization client and server, respectively, where the client initiates one offset measurement every second. To evaluate the effects of a volumetric DDoS attack, a load generator in AS_C aims to overwhelm the border router in AS_B, in an attempt to create asymmetric delay or cause packets even to be dropped. The entire topology and software can be deployed using a simple, self-contained orchestrator application. The orchestrator also allows to start and stop the attack traffic on the load generator, to en- or disable DOSC, and to gather and visualize the measured time offsets. We took precautions to not harm any real network infrastructure: Our experiments were directed at specific benchmarks and small in scale. We ensured that traffic only targeted our own instances and did not escape the AWS network into the public Internet.

Evaluation. The results of the offset measurements, with



(a) Software-based router with custom time server and client.



(b) Linux-based router with offset measurements using chrony.

Figure 8: Time offset measurements between two neighboring ASes with and without DOSC enabled. The border router in AS B is under attack during an interval of one or two minutes, respectively (interval highlighted in red).

and without DOSC enabled, are illustrated in Figure 8. As expected, in both cases we observe low time offset values during attack-free periods. However, once the DDoS traffic generation at the load generator gets triggered, in the non-DOSC-enabled setting the offset increases to over 21 ms. This effect is due to one of the border router egress queues growing (the one targeted by the load generator), increasing the latency for the time synchronization traffic in one direction, thus causing asymmetric latencies. In contrast, with DOSC enabled, time synchronization traffic is always forwarded in both directions without suffering increased queuing latency despite the volumetric DDoS attack, resulting in a time offset which is two orders of magnitude lower than without DOSC in place.

The maximum queuing latency is determined by the router's egress queue length, forwarding speed, and the link's capacity. Thus, longer paths are more vulnerable, as latency can increase with each on-path border router, potentially in one direction only. Reducing queue length may alleviate asymmetric latency but also raises the risk of time synchronization packets being dropped. In practice, network operators typically require router manufacturers to provide 250ms (or more) of buffering [43], where this value is frequently exceeded [44]. With WNB, synchronization happens only between neighboring nodes, which already renders volumetric DDoS attacks more difficult—combined with DOSC, the attacks can be effectively mitigated.

7. Discussion and Future Work

Path Transparency and Integrity. While multi-hop inter-domain communication paths are not actively utilized for time synchronization in WNB (as time synchronization traffic is constrained to one hop), they still play a crucial role in computing the weights assigned to neighboring ASes. Consequently, WNB requires some level of path transparency and path integrity. First, for a path to a specific destination, the source time server must be able to determine the first on-path AS, i.e., the correct neighboring AS. Achieving this property is straightforward, as for any destination, the next hop is readily provided by the routing protocol and might vary over time. Second, the integrity of the forwarding path needs to be protected. If attackers manage to manipulate forwarding paths, they can influence the computed weights, thereby controlling which neighbors an AS synchronizes with. While protecting the integrity of the first on-path AS is of paramount importance, the integrity of all other ASes on the path should also not be neglected. Depending on the source's path selection strategy (e.g., shortest path), a successful attacker can add or remove ASes from the path, affecting the computed weights. Route integrity can be implemented through AS signatures such as provided by route origin attestation (ROA) and BGPSec, or by future Internet architectures such as SCION [45].

Real-world Deployment. We are currently working on rolling out attack-resistant time synchronization in A2A mode as part of a pilot initiative involving multiple independent networks across two countries. The A2A mode facilitates testing even without direct peering between participants. As mentioned in Section 4.2, switching to WNB can occur incrementally with each new peering connection. Efforts to recruit additional participants for this pilot project are ongoing.

8. Related Work

Today's time synchronization deployments are often confined to geographically restricted areas. While such local synchronization can reduce clock skew for some applications, others need synchronization across larger regions including multiple countries. However, existing solutions are often not scalable, too expensive, too complex, not secure in large networks with potentially malicious nodes, or require centralized trust, thus introducing single points of failure.

Local Synchronization. In datacenters, DTP [46] provides nano-second-precision synchronization at the physical layer but requires hardware upgrades. Huygens [47] achieves similar precision without them. Due to the sensitivity of GNSS to interruptions, some regulators mandate GNSS-independent time synchronization. Net Insight's solution [9], designed for 5G and critical networks, enables synchronization over existing, non-PTP infrastructure. Redundant reference clocks can be deployed at separate locations, ensuring continuity if the master clock fails. Under NAVISP Element 1 [48], GMV [49] evaluates combining a sparse GNSS station network with time distribution via protocols like NTP or Dynamic synchronous Transfer Mode (DTM), demonstrating accurate synchronization over hundreds of kilometers [50]. This approach requires centralized trust and may need additional security to ensure synchronization availability and precision. In wireless sensor networks,

unlike our inter-domain setting, the Gradient Time Synchronization Protocol (GTSP) [51] uses Gradient Clock Synchronization [52], where nodes synchronize their clocks with neighbors by correcting their time based on average offsets. However, this method is vulnerable to malicious participants. PTPSec [53] mitigates on-path delay attacks by sending synchronization messages over redundant paths.

Global Synchronization. Many global time synchronization solutions require trusting a single system or operator for scalability, creating a single point of failure.

A common approach is to equip servers with GNSS receivers for external clock synchronization. However, this can be unsuitable in areas with weak GNSS signals, and makes deployment complex and inflexible. In the hierarchical NTP architecture [54], only primary time servers synchronize with GNSS, while stratum n servers obtain their time from stratum $n-1$ servers. Recently, there has been growing recognition that the NTP architecture must also be resilient to Byzantine faults [55]. CesiumSpray [56] proposes deploying GPS receivers in local area networks (LANs), allowing hosts to synchronize time with nearby GPS servers. This approach reliably avoids path asymmetries common in inter-domain time synchronization. The Satellite Time and Location (STL) system [57] is a time reference using low Earth orbit (LEO) satellites. Compared to GPS, STL offers stronger signals, reduced susceptibility to degradation, and lower disruption potential. One recently proposed solution, G-SINC [18], closely aligns with our requirements as a distributed protocol among core ASes in the public Internet. It is robust against faulty, misconfigured, unavailable, or malicious participants and maintains low skew during GNSS failures. However, unlike Everdeen, G-SINC is closely tied to the SCION future Internet architecture [45], relying on its path symmetry, transparency, control, and trust root configurations for managing membership. As an A2A instantiation, G-SINC's paths traverse multiple ASes, complicating the protection of time synchronization traffic from congestion-related issues. While Everdeen performs well with low-cost commodity clocks, G-SINC relies on more expensive oscillators with a maximum drift of $10\,\mathrm{ms}$ per year.

Protection against Congestion. Existing solutions to protect time synchronization traffic from congestion, especially during volumetric DDoS attacks, are either costly, reactive, vulnerable to source address spoofing, ineffective against legitimate-looking attack traffic, or suffer from circular dependencies. Securing traffic over longer forwarding paths is particularly difficult due to inter-AS coordination challenges, source address spoofing, and broader trust assumptions. While NTP vulnerabilities and NTP amplification attacks have been widely studied [20], [21], [58], [59], little work addresses protecting time synchronization traffic from volumetric DDoS. Some solutions filter outliers during brief congestion [60], while others assume core infrastructure is never congested [61].

General DDoS protections are often unsuitable for time synchronization traffic. Network scrubbing introduces latency and jitter, while leased infrastructure is inflexible, centralized, costly, and limited in scalability. Secure inter-domain bandwidth reservation systems [62], [63], [64], [65] protect against DDoS but are unsuitable for time synchronization traffic due to their reliance on synchronized clocks, creating a cyclic dependency: desynchronized clocks invalidate reservations and thus protection against congestion. Everdeen breaks this cycle with DOSC, which does not require synchronized time, enabling robust global time synchronization on which bandwidth reservation systems can reliably be deployed.

DDoS solutions at routers include history-based IP filtering [66], fairness enforcement for traffic flows [67], [68], and attack mitigation on programmable switches [32], [69]. However, these reactive measures detect and classify attacks post-event, often failing against spoofed IP packets and offering no fundamental protection. In contrast, DOSC operates proactively, providing instantaneous protection. It effectively mitigates (the impact of) address spoofing and ensures low-latency forwarding regardless of adversary attack patterns, including pulse-wave [70] and botnet attacks [71].

9. Conclusion

Global time synchronization plays a central role in ensuring the reliability and performance of many critical networked applications, and there is growing recognition that these systems must enhance their robustness, particularly by reducing dependence on GNSSes.

To address the challenges associated with existing methods of global time synchronization—such as high costs, limited scalability, centralization, and lack of security—we introduce Everdeen. Everdeen reduces reliance on GNSSes by incorporating high-priority network-based synchronization, cuts costs by leveraging existing infrastructure, achieves scalability by exclusively synchronizing among neighboring ASes, and operates in a fully decentralized manner. Most importantly, Everdeen represents a significant step forward in security. While it offers provably stronger Byzantine fault tolerance than existing systems—an incremental but valuable advancement—its key innovation is robustness against network congestion, including large-scale DDoS attacks. This achievement is particularly surprising, given that time synchronization protocols are inherently vulnerable to congestion, and defending against DDoS attacks in globally distributed deployments, where no single operator controls the entire infrastructure, is notoriously difficult.

Beyond our ongoing efforts to collaborate with parties interested in our testbed deployment, a promising direction for future research is to extend our DDoS defense mechanism to protocols beyond time synchronization. This approach is especially well-suited for critical protocols that rely on (or can be adapted to rely on) neighbor-based communication with minimal bandwidth requirements. By broadening the scope of our solution, we aim to pave the way for highly available critical communication.

Acknowledgments

We wish to express our gratitude to Juan Angel García-Pardo, Francesco Da Dalt, and Christoph Lenzen for

their feedback on the manuscript. We also thank Seyedali Tabaeiaghdaei for providing us with the CAIDA 2000 topology. We appreciate the financial support received from ETH Zurich, the Zurich Information Security and Privacy Center (ZISC), and armasuisse Science and Technology. Additionally, we gratefully acknowledge support for this project from the Werner Siemens Stiftung (WSS) Centre for Cyber Trust at ETH Zurich.

References

[1] G. O. for Science, "Satellite-derived time and position: Blackett review," https://tinyurl.com/3hj73daz, 2023.

[2] M. A. Lombardi, "An Evaluation of Dependencies of Critical Infrastructure Timing Systems on the Global Positioning System (GPS)," https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.2189.pdf, Tech. Rep., 2023.

[3] L. Economics, "Economic impact to the UK of a disruption to GNSS," https://tinyurl.com/yjfchn2e, 2023.

[4] J. Marshall, "Project PowerUp – Helping to keep the lights on in Ukraine in the face of electronic warfare," https://blog.talosintelligence.com/project-powerup-ukraine-grid/, 2023.

[5] B. Hubert, "GPS, galileo & more: How do they work & what happened during the big outage?" https://berthub.eu/articles/posts/gps-gnss-how-do-they-work/, 2023.

[6] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley, "GPS Software Attacks," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2012.

[7] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the Requirements for Successful GPS Spoofing Attacks," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2011.

[8] S. A. Jyothi, "Solar Superstorms: Planning for an Internet Apocalypse," in *Proceedings of the ACM SIGCOMM Conference*, 2021.

[9] Net Insight, "GNSS independent synchronization solution with open and disaggregated infrastructure," https://tinyurl.com/mr3k5h39, 2023.

[10] ——, "Keeping pace without satellites," https://netinsight.net/keeping-pace-without-satellites/, 2024.

[11] J. A. Sherman, L. Arissian, R. C. Brown, M. J. Deutch, E. A. Donley, V. Gerginov, J. Levine, G. K. Nelson, A. N. Novick, B. R. Patla, T. E. Parker, B. K. Stuhl, D. D. Sutton, J. Yao, W. C. Yates, V. Zhang, and M. A. Lombardi, "A resilient architecture for the realization and distribution of Coordinated Universal Time to critical infrastructure systems in the United States: Methodologies and recommendations from the National Institute of Standards and Technology (NIST)," https://doi.org/10.6028/nist.tn.2187, 2021.

[12] M. Ullmann and M. Vögeler, "Delay attacks — Implication on NTP and PTP time synchronization," in *Proceedings of the International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2009.

[13] R. Annessi, J. Fabini, and T. Zseby, "It's about Time: Securing Broadcast Time Synchronization with Data Origin Authentication," in *Proceedings of the International Conference on Computer Communication and Networks (ICCCN)*, 2017.

[14] Y. Perry, N. Rozen-Schiff, and M. Schapira, "A Devil of a Time: How Vulnerable is NTP to Malicious Timeservers?" in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2021.

[15] M. Wyss, M. Frei, J. Kwon, and A. Perrig, "Low-cost and robust global time synchronization," 2025.

[16] Wilfried Steiner, "State-of-the-art fault-tolerant clock synchronization for ultra-high reliable systems," https://tinyurl.com/bdkyhfww, 2024.

[17] P. Khanchandani and C. Lenzen, "Self-stabilizing byzantine clock synchronization with optimal precision," *Theory of Computing Systems (TOCS)*, 2019. [Online]. Available: https://doi.org/10.1007/s00224-017-9840-3

[18] M. Frei, J. Kwon, S. Tabaeiaghdaei, M. Wyss, C. Lenzen, and A. Perrig, "G-SINC: Global Synchronization Infrastructure for Network Clocks," in *Proceedings of the International Symposium on Reliable Distributed Systems (SRDS)*, 2022.

[19] G. Schmid, "Thirty years of dns insecurity: Current issues and perspectives," *IEEE Communications Surveys & Tutorials*, 2021.

[20] Cloudflare, "NTP amplification DDoS attack," https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/, 2024.

[21] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? reducing the impact of Amplification DDoS attacks," in *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, 2014. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kuhrer

[22] K. Wüst and A. Gervais, "Do you Need a Blockchain?" in *Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018.

[23] S. Blake, D. Black, M. Carlson, E. B. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," IETF RFC 2475, 1998.

[24] D. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, "Network Time Security for the Network Time Protocol," IETF, RFC 8915, Sep. 2020. [Online]. Available: http://tools.ietf.org/rfc/rfc8915.txt

[25] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight Source Authentication and Path Validation," in *Proceedings of the ACM SIGCOMM Conference*, 2014.

[26] B. Rothenberger, D. Roos, M. Legner, and A. Perrig, "PISKES: Pragmatic Internet-Scale Key-Establishment System," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2020.

[27] A. Custura, G. Fairhurst, and R. Secchi, "Considerations for Assigning a New Recommended Differentiated Services Code Point (DSCP)," RFC 9435, Jul. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9435

[28] Cisco, "Implement Quality of Service Policies with Differentiated Services Code Point," https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking/10103-dscpvalues.html, 2024.

[29] Juniper, "Class of Service User Guide," https://www.juniper.net/documentation/us/en/software/junos/cos/topics/example/cos-ba-classifier-dscp-default.html, 2024.

[30] Huawei, "Overview of Traffic Policing, Traffic Shaping, and Interface-based Rate Limiting," https://tinyurl.com/bdcdkjz5, 2024.

[31] CISCO, "QoS: Policing and Shaping Configuration Guide," https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_plcshp/configuration/12-4t/qos-plcshp-12-4t-book/qos-plcshp-oview.html, 2012.

[32] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Network and Distributed System Security Symposium (NDSS)*, 2020.

[33] M. Wyss and A. Perrig, "Zero-setup intermediate-rate communication guarantees in a global internet," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/wyss

[34] L. Brown, A. G. Alcoz, F. Cangialosi, A. Narayan, M. Alizadeh, H. Balakrishnan, E. Friedman, E. Katz-Bassett, A. Krishnamurthy, M. Schapira, and S. Shenker, "Principles for internet congestion management," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. ACM SIGCOMM '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3651890.3672247

[35] C. for Applied Internet Data Analysis, "AS Relationships," https://www.caida.org/catalog/datasets/as-relationships-geo/, 2023.

[36] A. D. Broido and A. Clauset, "Scale-free networks are rare," *Nature communications*, 2019.

[37] G. LLC, "The Go programming language," https://go.dev/, 2023.

[38] A. Najafi and M. Wei, "Graham: Synchronizing Clocks by Leveraging Local Clock Properties," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2022.

[39] Cisco, "QoS: Policing and Shaping Configuration Guide," https://tinyurl.com/yscpsvpm, 2023.

[40] Juniper, "Class of service user guide," https://tinyurl.com/ydpbjdhc, 2023.

[41] B. Hubert, *tc(8) — Linux manual page*, https://man7.org/linux/man-pages/man8/tc.8.html, 2001.

[42] R. Curnow and M. Lichvar, "Chrony," https://chrony-project.org/, 2023.

[43] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *ACM SIGCOMM Computer Communication Review*, 2004.

[44] Google, "BBR v2: A model-based congestion control," https://tinyurl.com/25khc6et, 2023.

[45] L. Chuat, M. Legner, D. Basin, D. Hausheer, S. Hitz, P. Müller, and A. Perrig, *The Complete Guide to SCION*. Springer International Publishing, 2022.

[46] K. S. Lee, H. Wang, V. Shrivastav, and H. Weatherspoon, "Globally Synchronized Time via Datacenter Networks," in *Proceedings of the ACM SIGCOMM Conference*, 2016.

[47] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosunblum, and A. Vahdat, "Exploiting a Natural Network Effect for Scalable, Fine-Grained Clock Synchronization," in *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2018.

[48] E. S. Agency, "ESA's NAVISP Element 1," https://navisp.esa.int/element/innovation, 2023.

[49] GMV, "GMV innovating solutions," https://www.gmv.com/, 2023.

[50] ——, "NAVISP element 1: Resilient, trustworthy, ubiquitous time transfer," https://tinyurl.com/4atn3ue8, 2023.

[51] P. Sommer and R. Wattenhofer, "Gradient Clock Synchronization in Wireless Sensor Networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2009.

[52] R. Fan and N. Lynch, "Gradient Clock Synchronization," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2004.

[53] A. Finkenzeller, O. Butowski, E. Regnath, M. Hamad, and S. Steinhorst, "PTPsec: Securing the precision time protocol against time delay attacks using cyclic path asymmetry analysis," in *IEEE INFOCOM*, 2024.

[54] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," IETF, RFC 5905, Jun. 2010. [Online]. Available: http://tools.ietf.org/rfc/rfc5905.txt

[55] N. R. Schiff, M. Schapira, D. Dolev, and O. Deutsch, "Preventing (network) time travel with chronos," in *Proceedings of the Applied Networking Research Workshop*, ser. ANRW '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3232755.3232766

[56] P. Veríssimo, L. Rodrigues, and A. Casimiro, "Cesiumspray: A Precise and Accurate Global Time Service for Large-scale Systems," *Real-Time Systems*, 1997.

[57] Satelles, "STL for time synchronization," https://satelles.com/technology/stl-complements-gps/, 2023.

[58] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg, "Attacking the network time protocol," Cryptology ePrint Archive, Paper 2015/1020, 2015, https://eprint.iacr.org/2015/1020. [Online]. Available: https://eprint.iacr.org/2015/1020

[59] T. Mizrahi, "Security Requirements of Time Protocols in Packet Switched Networks," IETF, RFC 7384, Oct. 2014. [Online]. Available: http://tools.ietf.org/rfc/rfc7384.txt

[60] D. Veitch, J. Ridoux, and S. B. Korada, "Robust synchronization of absolute and difference clocks over networks," *IEEE/ACM Transactions on Networking*, 2009.

[61] M. Rossberg, R. Golembewski, and G. Schaefer, "Attack-resistant distributed time synchronization for virtual private networks," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, 2012.

[62] C. Basescu, R. Reischuk, P. Szalachowski, A. Perrig, Y. Zhang, H.-C. Hsiao, A. Kubota, and J. Urakawa, "SIBRA: Scalable Internet Bandwidth Reservation Architecture," in *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2016.

[63] G. Giuliari, D. Roos, M. Wyss, J. A. García-Pardo, M. Legner, and A. Perrig, "Colibri: A Cooperative Lightweight Inter-domain Bandwidth-Reservation Infrastructure," in *Proceedings of the Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2021.

[64] M. Wyss, G. Giuliari, M. Legner, and A. Perrig, "Secure and Scalable QoS for Critical Applications," in *Proceedings of the IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2021.

[65] M. Wyss, G. Giuliari, J. Mohler, and A. Perrig, "Protecting Critical Inter-Domain Communication through Flyover Reservations," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[66] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based ip filtering," in *IEEE International Conference on Communications (ICC).*, 2003.

[67] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998. [Online]. Available: https://doi.org/10.1145/285237.285273

[68] Z. Yu, J. Wu, V. Braverman, I. Stoica, and X. Jin, "Twenty years after: Hierarchical core-stateless fair queueing," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/yu

[69] Z. Liu, H. Namkung, G. Nikolaidis, J. Lee, C. Kim, X. Jin, V. Braverman, M. Yu, and V. Sekar, "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches," in *30th USENIX Security Symposium*, 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/liu-zaoxing

[70] A. G. Alcoz, M. Strohmeier, V. Lenders, and L. Vanbever, "Aggregate-based congestion control for pulse-wave ddos defense," in *Proceedings of the ACM SIGCOMM Conference*, ser. SIGCOMM, 2022.

[71] Cloudflare, "Cloudflare DDoS threat report for 2023 q4," https://blog.cloudflare.com/ddos-threat-report-2023-q4, 2024.

[72] J. L. Welch and N. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization," *Information and Computation*, 1988.

Algorithm 1: GNSS-based Clock Synchronization

Input:
J — GNSS synchronization interval
C_{Sat} — Max. GNSS correction

```
1  clock.SetTime(GNSS.Time())
2  while true do
3      gnssTime = GNSS.Time()
4      localTime = clock.AdjustedTime()
5      δ = gnssTime - localTime
6      if |δ| > 0 then
7          corr = sgn(δ) · min(|δ|, C_Sat)
8          clock.AdjustTime(corr, J)
9      end
10     clock.Sleep(J)
11 end
```

Algorithm 2: Network-based Clock Synchronization

Input:
I — Network synchronization interval
C_{Net} — Max. Network correction
X — Network cutoff
P — Set of neighboring synchronization peers

```
1  n = |P|
2  while true do
3      M = [0]
4      for p ∈ P do
5          m = offset(clock.AdjustedTime(), p);    // Measure
             time offset
6          for i = 0; i < p.Weight; i++ do
7              M = M + [m]
8          end
9      end
10     M = sort(M)
11     δ = (M[n/3] + M[2n/3]) / 2;        // Fault-tolerant
         midpoint function tolerating up to F
         faults (N = 3F + 1)
12     if |δ| > X then
13         corr = sgn(δ) · min(|δ|, C_Net)
14         clock.AdjustTime(corr, I)
15     end
16     clock.Sleep(I)
17 end
```

Appendix

1. Everdeen's Algorithms

Global time synchronization among ASes is achieved through two algorithms: one synchronizing via GNSS signals and the other through neighbor-based Internet synchronization at each AS's time server.

GNSS-based Clock Synchronization. In Algorithm 1, a time server periodically synchronizes with GNSS signals at intervals of duration J for precise, low-skew time synchronization. Any time deviation is gradually corrected over the next interval to avoid abrupt shifts, with corrections capped by a constant C_{Sat}.

Network-based Clock Synchronization. In Algorithm 2, every time server runs time offset measurements with the time servers of neighboring ASes at regular intervals of duration I. The algorithm gathers all the measured offsets from these peers into an array, sorts the array, and then applies a fault-tolerant midpoint function (here instantiated with the function proposed by Lynch and Welch [72]). The output of this function is used to uniformly adjust the AS' clock during the subsequent synchronization interval. This correction is bounded by the constant C_{Net}. It is only applied if it is larger than the network cutoff X. A value for X slightly above the network synchronization error ensures that during normal operation when GNSSes are available, Everdeen achieves the same precision as when solely relying on GNSSes, i.e., without network-based synchronization. The offset measured for a neighbor is added multiple times based on the weight assigned to that neighbor, as shown in Line 6 to Line 8 of Algorithm 2. Although this method is simple, it is inefficient. In Appendix B, we demonstrate how to remove this overhead.

Algorithm Coordination. Actual time retrieval from a server's clock is accomplished using `clock.Time()`. However, in the algorithms, we retrieve time as `clock.AdjustedTime()` (as indicated in Line 4 of Algorithm 1 and Line 5 of Algorithm 2). This method returns the current time with all fully applied time corrections. The reason for this is to prevent the algorithms from overcorrecting the time target. Such over-correction could otherwise occur when time is already closely synchronized with the GNSS value, and both algorithms independently decide to apply a correction, where a single correction would suffice to reach the GNSS value. Without using `clock.AdjustedTime()`, this issue would arise due to a lack of coordination between the two algorithms.

2. Midpoint Computation

A straightforward implementation of the midpoint computation in WNB maintains an array of the same size as in A2A. In this array, entries for non-neighboring nodes are duplicates of the first on-path node's entry (see Figure 2). This implementation results in a computation overhead equivalent to that of A2A. However, a more efficient approach involves computing the midpoint directly on a significantly smaller array that contains only measurements to direct neighbors, along with their respective weights. This optimization leads to a substantial reduction in processing overhead.

As demonstrated in Table 3, the overhead reduction is remarkable. For example, in a topology with 103 nodes, each node's midpoint computation in A2A takes 55.12 µs. In contrast, for a node with degree (d) of 10 and per-neighbor weight (w) of 102, the optimized WNB variant only requires 0.27 µs, corresponding to a speedup of 204. These speedup results are particularly relevant in scenarios involving very large topologies, low-end devices, or high-frequency time synchronization. In other cases, the midpoint computation as in A2A is likely fast enough for practical purposes.

3. CAIDA Graph Characteristics

Figure 9 shows characteristics of the CAIDA 2000 topology. Figure 9a depicts the general distribution of node degrees, which closely adheres to a power law, indicating that

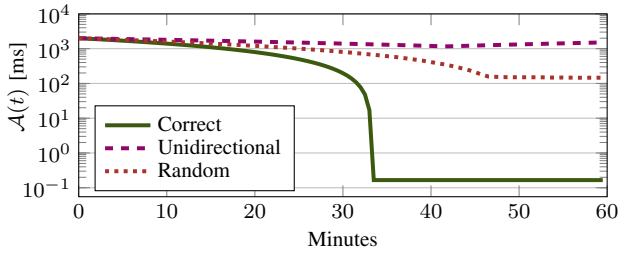


Figure 10: Everdeen's accuracy $\mathcal{A}(t)$ in milliseconds, both under correct operation and under incorrect GNSS values.

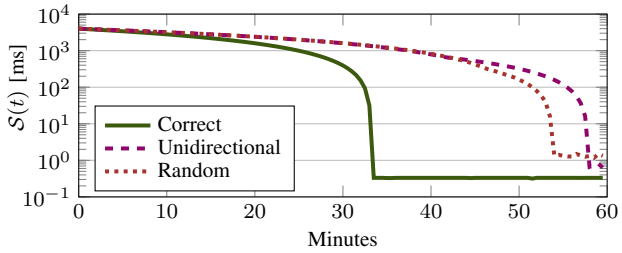
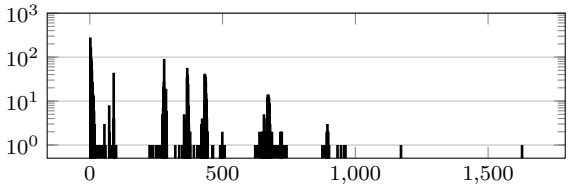


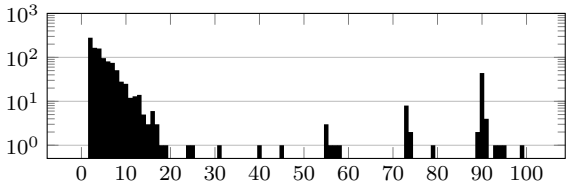
Figure 11: Everdeen's skew $\mathcal{S}(t)$ in milliseconds, both under correct operation and under incorrect GNSS values.

TABLE 3: Midpoint computation overhead in microseconds for A2A compared to the optimized midpoint algorithm for WNB. For the WNB evaluation, the weights w, the total number of nodes n, and the node degree d are related as $w \times d = n$.

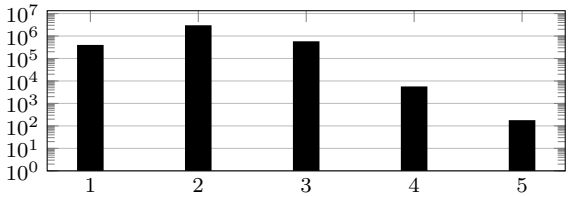
Total #ASes n	10^1	10^2	10^3	10^4
A2A	0.15	3.22	55.12	742.90
WNB, $w = 10^1$	0.08	0.27	4.27	70.56
WNB, $w = 10^2$	-	0.08	0.27	4.23
WNB, $w = 10^3$	-	-	0.08	0.27
WNB, $w = 10^4$	-	-	-	0.08



(a) Overall distribution of node degrees.



(b) Distribution of the lowest node degrees.



(c) Distribution of shortest path lengths.

Figure 9: Characteristics of the CAIDA 2000 topology.

the topology exhibits characteristics of a scale-free network. Figure 9b zooms in on the degree distribution of the lowest-degree nodes, revealing that the lowest node degree in this network is 2. Lastly, Figure 9c shows the distribution of the shortest path lengths between all pairs of nodes, where the maximum path length is 5.

4. GNSS Faults and Attacks

Everdeen ensures synchronized time with bounded skew, regardless of GNSS values, by prioritizing network-based synchronization over GNSS-based. Concretely, Everdeen implements this by requiring that in each synchronization interval $\mathtt{C}^{\mathtt{Net}} > \mathtt{D}+\mathtt{C}^{\mathtt{Sat}}$ (Section 5.4), meaning the maximum network-based correction exceeds the sum of the maximum GNSS-based correction and clock drift. Therefore, even if the clock drift and the GNSS-based synchronization maximally move the clock in one direction, the network-based synchronization can still overrule this decision and move the clock even in the complete opposite direction.

To verify this, we evaluate Everdeen under incorrect GNSS values, representing GNSS faults, misconfigurations, and attacks. We consider two particular cases on the spectrum of possible GNSSes value distributions. For both cases, we consider the worst-case scenario in which GNSSes are affected globally, impacting all the nodes and not just a subset of them. In the first case, GNSS values are provided in an attempt to maximally move the clocks in one consistent direction (i.e., make the clocks move slower). In the second case, GNSS values are provided that aim at maximally desynchronizing the clocks by moving each in a different (random) direction.

The impact on accuracy is shown in Figure 10. As expected, accurate time synchronization is not possible given globally incorrect GNSS values. Figure 11 illustrates the impact on the clock skew. In the first case with unidirectional GNSS values, the skew increases from ~0.3 ms to ~0.6 ms, which is comparable to a scenario with complete GNSS outage (~0.7 ms, Section 5.4). In the second case with random GNSS values, the skew increases to ~1.3 ms. Importantly, while incorrect GNSS values cause degraded clock skew and slower convergence, the skew is still globally bounded, i.e., the clocks remain synchronized and do not drift apart.