# Attack-Resistant Distributed Time Synchronization for Virtual Private Networks

Michael Rossberg, Rene Golembewski, Guenter Schaefer
Ilmenau University of Technology, Germany
Email: {michael.rossberg, rene.golembewski, guenter.schaefer} [at] tu-ilmenau.de

***Abstract*—To securely exchange data over public networks, such as the Internet, organizations often utilize Virtual Private Networks (VPNs). However, relying on these potentially large overlay networks makes them vital targets for Denial-of-Service (DoS) attacks. Thus, recent approaches for VPN auto-configuration address DoS resistance by employing distributed management algorithms. Nevertheless, there is no satisfying solution for time synchronization within VPNs that is designed for resistance against DoS as well as internal attacks. For example, Network Time Protocol (NTP) relies on hierarchical structures, and cannot comply with DoS resistance. Thus, in this article we present a novel, fully distributed and fault tolerant time synchronization approach, which is designed to be transparently integrated in VPN gateways. Combining diffusion-based round-trip-synchronization with an internal attacker detection, the proposed mechanism is making a contribution to resilient VPN design. Simulation results reveal a robustness against rather powerful internal attackers.**

## I. Introduction

In order to exchange private information over untrusted networks, organizations commonly depend on VPNs to connect branch offices and field works. Fig. 1 shows a typical scenario with six private networks, forming a VPN. Inside of these overlay networks some important security goals are inherently realized by a deployed VPN protocol, e.g., IPsec. The communicating nodes authenticate each other and transfer data in a way that it is protected against eavesdropping as well as man-in-the-middle attacks. Additionally, many VPN nodes are audit-proof, to allow for post-mortem analysis in case of security incidents. However, to guarantee these security services, time synchronicity is an often neglected, yet important prerequisite. The validation of certificates and distributed logging of network events are two examples, for which synchronized clocks are needed.

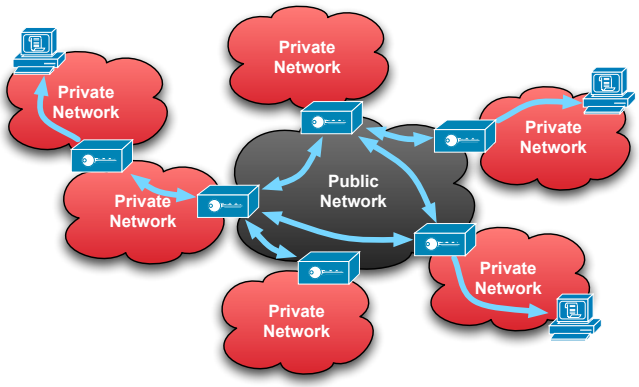



Fig. 1. VPN topology with nested subnets and distributed logging hosts

A disadvantage — especially of large VPNs — is the required configuration and maintenance effort, resulting in limited scalability with respect to number of subnetworks and difficulties in supporting mobile nodes. Hence, several auto-configuration approaches have been developed and at least partially been deployed over the last decade, and notably the more recent ones are organized in a fully distributed way to avoid single points of failure. Thus, VPNs can now be optimized with respect to the previously unaddressed DoS resistance. A fully automated system for a distributed VPN configuration is described in [1]. The approach proposes that all VPN gateways communicate via a peer-to-peer overlay structure with proactively established connections to scale with the number of nodes.

Yet, from the mentioned perspective of DoS resistance, it makes only little sense to design the VPN management algorithms in a distributed fashion, while other crucial higher layer functions like time synchronization, name resolution, or logging are still realized by central servers within the VPN. For example, the current de facto standard for time synchronization – NTP [2] – depends on a hierarchical server infrastructure. The goal is to achieve DoS-resilient behavior for the time synchronization service, and therefore construct a distributed algorithm. Due to the security sensitive environment of VPNs, the mechanism should also be able to resist internal attackers, which might compromise one or more gateways and subsequently try to disturb the synchronization process.

Thus, this article analyzes the current state of the art and presents a novel distributed approach for secure time synchronization with regard to internal as well as external attackers. Furthermore, we contribute an evaluation of convergence and attacker resistance by a simulation study.

The rest of the article is organized as follows: Sec. II describes the objectives for a distributed time synchronization in VPNs, which are then collated with the related work. In Sec. IV we describe the main system design and then describe the evaluation with respect to the objectives by analysis and simulation study in Sec. V. Finally, Sec. VI concludes and gives an overview of planned further research.

## II. Objectives

A time synchronization algorithm for the sketched scenario must meet the following requirements:

**1. Operation in global VPN environments:** As time needs to be adjusted within general Internet Protocol (IP) based VPNs, there may be no possibility to perform any-, broad-, or multicast [3]. Furthermore, the algorithms usually do not have access to link layer information.

**2. Internal clock synchronization:** As long as all VPN gateways are synchronized internally, the whole VPN is in a consistent state. Even though the clocks should be synchronized with the highest reachable precision, for VPNs a synchronization precision in the order of seconds is sufficient for the security services. For example, with such a mechanism the lifetime of certificates can correctly be verified and log files from different nodes can be easily correlated [4].

**3. Accordance of internal clocks and external time:** One consequence of a strict internal synchronization is a possible divergence of internal and external time. For reasons of robustness all core functions shall work properly even without a strict external synchronicity. However, the algorithm must support means to explicitly synchronize with external clock sources, whose time is securely set by administrative instances.

**4. Integrity:** Because of the encrypted VPN tunnels, external attackers are not able to undetectably change any transmitted data. Nonetheless, it is possible to drop or delay packets to gain influence on time synchronization. Especially the latter, which will be referred to as *delay attack*, is difficult to detect and must not lead to a manipulation of the synchronization mechanism. Besides, internal attackers are potentially able to take part in the synchronization process and may try to influence other nodes. The algorithm shall be robust to such attacks as long as not more than half of the neighboring nodes are compromised.

**5. DoS Resistance:** Even if one or more nodes are unable to communicate due to a DoS attack, the synchronization of the remaining ones must not be affected. Within this article we assume that the core network infrastructure is unaffected of the attack, because of its high capacity links [5], [6]. The effects of DoS attacks may result in congested links to some of the VPN gateways, resulting in high delay, packet loss and even entire failure.

**6. Robustness:** Related to DoS resistance is the robustness against random failures or harsh network conditions. If no communication between two or more groups of nodes is possible the VPN can be partitioned, e.g., due to broken links in the transport network. Within the partitions synchronization must be continued, and after reunion a synchronization of the previously disconnected parts must take place.

**7. Efficiency & Scalability:** The induced communication and calculation overhead shall be as small as possible. Furthermore in each peer, it shall grow at most logarithmically with the total number of peers, so that even large VPN installations can be deployed.

## III. Related Work

This section gives an overview over the state of the art in synchronizing clocks over communication networks. The most important approaches are described in more detail and rated according to the previously introduced objectives.

### A. Traditional Synchronization Approaches

NTP and the Simple Network Time Protocol (SNTP) [2], [7] are perhaps the most popular protocols for time synchronization in IP networks. Being hierarchical approaches, synchronization information is distributed from an external clock source at the top over intermediate servers to clients at the lowest level. Although actually multiple time servers can be used, DoS attacks and compromised nodes may cause serious problems. Thus, NTP is inappropriate in VPNs with high requirements on availability.

Further approaches for fixed networks are the well known Berkeley synchronization [8] and the similar algorithm of Cristian [9]. In both mechanisms time servers send out time values to the clients, which adjust their local clock without any further checks. These centralized servers are exposed targets for attacks, as it is possible to control all individual times by compromising the master.

### B. Synchronization Approaches without Exposed Nodes

A first approach to distributed time synchronization was described by Gurewitz et al. [4]. Therein, a mathematical optimization is used to minimize the synchronization errors towards a set of reference nodes. However, attackers and network partitioning are disregarded and the reference nodes pose a potential target for attackers. A system to overcome some limitations such as the need for symmetric delays by estimating reliable one-way delays was presented in [10]. Nonetheless, it requires at least two disjoint paths between any two nodes and does not discuss attacks.

More recently, distributed time synchronization has been discussed in the context of Wireless Sensor Networks (WSNs) [11], and when comparing VPNs and WSNs similarities in the failure model emerge. Both require fully distributed approaches to manage random failures and internal as well as external attackers. The authors of [12] and [13] summarize the most important algorithms for time synchronization in WSNs. Li and Rus propose a method called *Asynchronous Diffusion* [14] for use in WSNs, which depends on the asynchronous exchange of timestamps between neighbors. The obtained mean time is then adjusted to and periodically sent out to the neighbors, which in turn adjust their local clocks afterwards. To be robust against attacks, several nodes with a higher complexity and additional security features are used that shall be seen as “always trustful”. However, the use of link layer broadcasts and the trust model makes the algorithm not directly deployable in VPNs. Other approaches combine the model with linear regression [15], but the use of least mean square fitting makes the approaches even more vulnerable to malicious nodes.

Current research focuses on refining the approaches to realize attack resistance and to cope with node breakdowns. However, [16] does not take internal attackers into account. In [17] internal attackers are discussed, but the proposed outlier detection algorithm requires extensive parameter tuning, which is not possible in dynamic VPNs. The authors of [18] consider a purely hierarchical service, which is likely to suffer from DoS attacks and consistent lying of nodes of one level. The perhaps most interesting one is the diffusion-based approach [19], where timestamps are solely accepted if one node with a larger and one node with a smaller timestamp is found. However, neither slowly drifting attackers in combination with jitter nor collusions can be addressed.

All in all, basic principles of WSN synchronization are applicable to VPNs, but the attack resistance and dynamics require the development of novel techniques.

## IV. System Design

As outlined, a distributed synchronization of clocks can be achieved by the periodic exchange of timestamps and subsequent adjustments to the calculated offsets. In the following, we propose an own approach that is also based on this idea. However, due to filtering mechanisms not only outliers can be detected, but it is also possible to achieve a resilience against internal attackers. These filters are used at different phases of the algorithm and will be discussed in the following sections.

### A. Measurement of Time Synchronization Information

After an initial setup of frequency and timing information on the basis of the local hardware clock, the periodic neighborhood synchronization starts. Its main principle is the periodic measurement of Round-Trip-Times (RTTs) and time differences between neighboring nodes according to the protocol depicted in Fig. 2. All messages are expected to be integrity protected and authenticated by the underlaying VPN. In order to assure the freshness of the measurements, a request/reply scheme is used, in which all queries are immediately answered including a nonce from the request. To avoid delay attacks of arbitrary lengths, nonces will be marked invalid after a short, reasonable time, e.g., one second.

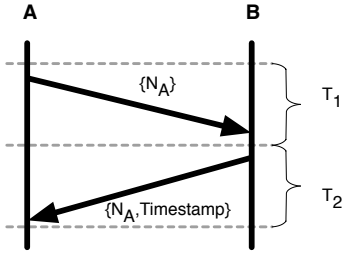



Fig. 2. Simple request/reply scheme to exchange timing information

In an undisturbed environment, nodes may calculate offsets to another node by adding $^{RTT}/_{2}$ to the reported timestamp and comparing it to their local time. However, already in Internet-like environments without attackers this is not always the case, due to $T_1$ and $T_2$ not being equal. Among others [20], [4] state that these measurements are particularly subject to:

- *Jitter:* The transferring delays of request and reply may vary independently, e.g., due to retransmissions or scheduling issues. As outlined in [21] jitter can be described by a Cauchy process. Thus, many of the measurement messages will have no additional delay, while others may be delayed much longer.
- *Queueing Delays:* In the case of congestion, the queueing delays of both packets may increase independently. This may also affect a single direction of the data flow, so that $T_1$ and $T_2$ are unequal.
- *Asymmetric Paths:* In some setups there may be different paths used for sending and receiving data, e.g., when data is received via a broadband satellite downlink. In this case there may be a long-term asymmetric delay, which prevents a good calibration.

However, while a calibration in the presence of the short-term asymmetric effects is possible, long-term asymmetries might not be detectable by a time synchronization algorithm. When the asymmetric delay equally affects all neighboring nodes, the node that is connected via the asymmetric link will adjust to a different time. However, this time will appear correctly for all participants as also the measurements from other nodes are based on two-way measurements and thus be affected by the asymmetry. Without further assumptions, e.g., two independent uplinks per node, a synchronization below half of the RTT error is impossible and remains out of scope of this article.

In security sensitive environments the situation becomes even more difficult: It is not sufficient to only regard erratic short- and long-term effects. Instead an malicious influence by one or more of the following attack schemes must be regarded:

- *DoS attacks* may cause an extreme congestion that leads to high asymmetric delays. Additionally, the link is subject to high packet-loss and jitter.
- *Delay Attacks:* Attackers with access to the communication path may furthermore arbitrarily delay synchronization messages. Depending on the strategy, they may try to influence the measurement by adding either highly asymmetric or noisy delays.
- *Internal Attackers:* Perhaps the most influence can be obtained by internal attackers. Additionally to the capabilities of a delay attacker, they may change their reported timestamps and offsets arbitrarily.

All in all, the measured RTTs and timestamp information may be subject to various distortions, which makes direct adjustment difficult. Thus, the following section will discuss several mechanisms to detect outliers.

### B. Outlier Detection

In a first step, every node collects the offsets and the RTTs in two vectors for each neighboring node, as outlined in Alg. 1. Once there are enough measurements, e.g. 10 for each neighbor, the vectors are used to estimate the true offsets without any short term effects, i.e., jitter.

The estimates are obtained by a Repeated Median Estimator (RME) [22], which performs a robust linear regression. Due to the nature of the RME, the regression is guaranteed to be unbiased regardless of the distribution of distortions as long as not more than half of the values are erroneous. Thus, it allows not only for a robust estimation of the undistorted values, but also for their change over time, e.g., the drift of other nodes.

**Algorithm 1:** Offset estimation for neighbor $n$

```
   Input: n, offset, rtt
1  offsets[n].insert(offset);
2  rtts[n].insert(rtt);
3  if |offsets[n]| < minOffsets then
4      return (0, 0);
5  rmeOffset ← RME(offsets[n]).valueAt(now);
6  estOffset ← min_abs(offset, rmeOffset);
7  confidence ← (1 + RME(offsets[n])).slope)^-1
8                · (1 + RME(rtts[n]).slope)^-1
9                · (1 + (maxOffsets − |offsets[n]|)/maxOffsets)^-1;
10 if |offsets[n]| ≥ maxOffsets then
11     r ← zipfrand(N=|offsets[n]|, s=1);
12     offsets[n].erase(r);
13     rtts[n].erase(r);
14 return (estOffset, confidence);
```

Depending on the number of already collected values, these estimates become more accurate in terms of sensitivity against jitter. Furthermore, a longer history also prohibits a quick change of the estimated offsets if an attacker is able to influence the offsets, e.g., due to reporting wrong timestamps or DoS attacks. In this case, the newer values will be treated as outliers and the offset will still be estimated correctly. However, the estimator slows down adjustments to legitimate time changes, e.g., when the time converges in the network. Thus, the proposed algorithm uses either the currently measured offset or the estimate of the RME for further calculations, which ever has the smaller absolute value (line 6).

Furthermore, for each estimation a confidence value in the interval of $(0..1]$ is calculated. In particular the confidence can be reduced by three factors:

- the estimated slope of the offset to the other node, which indicates a random or malicious time drift,
- the change in the RTTs, e.g., due to mobility or instabilities in the network, and
- the amount of free space in the sample vector, which allows to mistrust nodes that only participated for a short time.

The factors are multiplied with equal weight as a violation of each of the factors may equally degenerate the estimate. Since the factors are independent, the multiplicative effect ensures that the reported confidence will decrease much faster, if more than one factor indicates an instability.

In order to restrict the growth of the vectors, an element is removed once they reach a maximal desired size, e.g., 100 elements. This may be especially important as the RME has a computational complexity of $\mathcal{O}(n^2 \log n)$ and is calculated rather often. As the observable time frame, and thus the time to detect long-term effects, shortens when the naïve method of removing the oldest element is used, the removal occurs by deleting a random element. Due to the usage of a Zipf distribution for the selection, newer entries are more likely deleted and thus a longer time span can be evaluated. Nonetheless, having a positive probability of being deleted, also the older values get eventually thinned out, and thus their influence fades over time. Depending on the parameter $s$ more or less old values will be preferred in the process. However, in accordance with the intended conservativeness of the process and maximum vector size, values between $[0.5, 2.0]$ seem to be reasonable. Due to the mechanism, a vector of 100 elements typically covers between 50,000 ($s = 0.5$) and 1,000 ($s = 2.0$) synchronization steps, which can be obtained by simple simulations for example.

After calculating estimates and confidence values for each neighbor individually, the values will be put in relation to each other, which is covered in the next section.

### C. Adjusting System Time

Using the estimated offsets, it is the principle idea of distributed time synchronization mechanisms to adapt the own clock towards these offsets. While other approaches suggest to average the offsets [23] or minimize the quadratic error [10] of their embedding, our system utilizes the median function. Thus, if the attackers are able to compromise just a few nodes, which is the expected case in VPNs, they are not able to influence the rest of the nodes at all. Even if more nodes should be compromised the influence is considered to stay low.

**Algorithm 2:** Adjustment of System Time

```
   Input: estOffsets, confidences
1  medOffset ← wMedian(estOffsets,confidences);
2  if medOffset < minAdjOffset then
3      return;
4  adjustTime(medOffset · dampFactor);
5  foreach neighbor n do
6      offsets[n].adjustBy(medOffset · dampFactor);
```

The actual adjustment is performed according to Alg. 2. The average offset is calculated by a weighted median function, which prefers reported offsets with higher confidence values (line 1). If this average is above the desired accuracy an adjustment is performed. However, in order to prevent oscillation effects a dampening factor, such as 0.1, is applied first, so that time changes are performed gradually. In a last step (line 6) the previously obtained measurements are adjusted, as they still refer to the old time.

### D. Requirements for Secure and Robust Operation

While the introduction of the weighted median gives a certain resistance even against internal attackers, further precautions have to be taken at VPN layer. In particular, the VPN overlay has to ensure a resistance against typical peer-to-peer-related attacks, such as Sybil [24] and Eclipse [25] attacks. Otherwise, a single attacker might increase its weight in the synchronization process by introducing additional nodes, or by virtually surrounding honest nodes. While the introduction of new nodes may be countered by the usage of signatures and certified keys, the latter is more difficult to prevent. Especially as the attacks let the surrounded nodes adapt to the attacker's time and therefore transport this view also to other nodes. Therefore, the time synchronization should only be performed with nodes that verifiably did not initiate the security association to just spread their timing information. This can be achieved by two measures: First, time synchronization of nodes can be taken into account, if the security association was initiated locally. Second, if it was established due to verifiable overlay topology constraints, e.g., between ring neighbors in a Chord ring with authenticated node IDs.

A second prerequisite on the overlay topology affects the convergence: Due to the outlier detection and the usage of the median, the convergence proofs of other approaches, e.g., for the directed diffusion [23], are not applicable. In fact:

**Lemma.** *With the simplification of equal confidence values, a convergence can only be guaranteed if and only if the following condition holds:*

$$\Big\{\forall P \subset V \;\Big|\; |P| > \frac{|V|}{2} \vee \exists\, v_p \in P : \Big|\bigcup_{w \notin P} e(v_p, w)\Big| < \Big|\bigcup_{w \in P} e(v_p, w)\Big|\Big\}$$

Thus, there must be no partition $\tilde{P}$ that is:

1) less than half of the network size $|V|$ and
2) contains a node $v_p$ that has more synchronization neighbors within $\tilde{P}$ than within $V \setminus \tilde{P}$, i.e., $\tilde{P}$ is a community structure.

*Proof:*

1) $\exists \tilde{P} \Rightarrow$ Synchronization may fail:
   If a partition $\tilde{P}$ with the aforementioned properties exists, its nodes $v_{\tilde{p}}$ might have a consistent internal synchronization. As all of these $v_{\tilde{p}}$ have more neighbors within the partition and the median is stable against up to 50% outliers, none of them will adapt to the rest of the network. If the rest of the overlay is also having a consistent internal synchronization, it might not adapt to the time in $\tilde{P}$ either and an overlay wide synchronization fails.
2) $\nexists \tilde{P} \Rightarrow$ Synchronization is guaranteed: If no $\tilde{P}$ exists, then all partitions $P$ must either contain a suitable node $v_p$ or must be larger than $|V|/2$.
   a) If $P$ is small enough, but at least one node $v_p$ has more synchronization neighbors outside of $P$, then $v_p$ will synchronize with the outside offsets due to the median. Hence, no stable inner synchronization is possible within $P$. Therefore, either the rest of $P$ synchronizes to $\{v_p\} \cup V \setminus P$ or a non-empty partition $P_2 \subseteq P \setminus v$ exists. However, as $P_2$ must be even smaller than $P$ and it must contain no node, which is more densely connected to the outside, as this would contradict to the presumption.
   b) If $P$ exists, but is larger than $|V|/2$, then either all nodes $V \setminus P$ synchronize with $P$ or a second community $P_2$ must exist. If $P_2 \cap P = \varnothing$ synchronization may fail, but would contradict to the presumption as $P_2$ would be smaller than $|V|/2$. If $P_2 \cap P \neq \varnothing$, $P_2$ must be either a sub- or superset of $P$. Without loss of generality $P_2$ is a subset of $P$ and all nodes $v \in P \setminus P_2$ would synchronize to the time in $P_2$.

■

All in all, the outlined partitions may lead to asynchronous VPN, if they are consistently synchronized internally. Thus, every node within the partition must synchronize only within the partition, and the rest of the network does not synchronize to the partition. Furthermore, these partitions must be persistent, i.e., it is not sufficient if they exist for a short period only. Physical network splits, e.g., due to a network failure, also form partitions in this sense, as they do not include links between each other, and at least one of the partitions is smaller or equal to $|V|/2$.

Depending on the actual structure of the network these convergence properties might pose a severe drawback. For example, in common router topologies or social graphs, areas that are more densely connected are rather common. However, the common, structured overlay topologies do not suffer from this effect. The major reason for this immunity is the effort of the designers to achieve a small network diameter with as few as possible links. Thus, links in more densely connected areas would be better placed to distant targets to achieve a lower diameter. In particular, at least for structures like CAN, Chord, Kademlia, Butterfly graphs, and entangled networks such partitions can easily be ruled out. Also, our VPN system [1] does not construct such partitions and therefore a synchronization can be guaranteed.

## V. Evaluation

In order to evaluate the presented algorithm, we discuss several aspects qualitatively with regard to the given objectives, as given in Sec. II. Quantitative measurements were performed in a very detailed simulation environment, which is based on OMNeT++ [26], INET, and an additionally implemented IPsec stack. In Internet scenarios, the deployed VPN overlay approach [1] establishes Chord-like topologies with the following properties (averaged over 32 runs):

| Nodes | Min/Avg/Max Degree | Diameter | Avg length of shortest path |
|---|---|---|---|
| 100 | 7.03 / 10.85 / 15.70 | 3.15 | 2.16 |

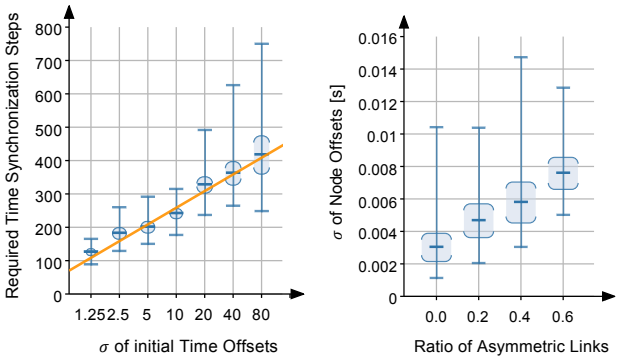
The delays within the transport network were taken from the Meridian Dataset [27], which represents Internet delays. However, due to the measurement method it contains neither jitter nor information on the asymmetry of paths. Thus, additional jitter has been added with a Cauchy distribution [21], e.g., with the parameter $\gamma = 0.25$ this results in an additional RTT delay that is usually below $1ms$. However, as for the Cauchy distribution no mean value exists, much longer delays, e.g., $25ms$, are common. Additionally, random links were chosen to have an asymmetric delay of up to 50%.
If not stated differently, the following parameter set is used:

| | | | |
|---|---|---|---|
| Nodes | 100 | minOffsets | 10 |
| Jitter | $\gamma = 0.25$ | maxOffsets | 100 |
| Clock Offsets | $\mathcal{N}(0, 10s)$ | minAdjOffset | 0.001 |
| Asymmetry | 10% of links | dampFactor | 0.1 |

The values in all of the following charts are means over 32 simulation runs given with a 99% confidence interval and min-max-whiskers.

**1. Operation in global VPN environments:** The algorithm does not make a use of any cross-layer information, nor any-, broad-, and multicast messages. Hence, it is deployable in generic IP networks.

**2. Internal clock synchronization:** To quantitatively evaluate the synchronization speed, we performed simulations, where all clocks had a normally distributed offset. This reflects a scenario, when all nodes are initially switched on, i.e., when creating an ad-hoc VPN. For standard deviations of the time offsets, the number of synchronization operations were measured until a synchronization of $0.1s$ was reached. Fig. 3(a) shows an about linear influence, on a logarithmically scaled x-axis. Thus, even if the nodes have wide spread time offsets a fast synchronization will take place. Besides the initial offsets, only the size of the VPN was found to have a significant influence on the synchronization speed, which will be discussed under the point *scalability*.




(a) Speed in dependence of initial clock deviation (b) Precision in dependence of asymmetry

Fig. 3. Basic synchronization properties

While the effects of jitter can be efficiently filtered, the effects of long-term asymmetries may have an impact on the achievable synchronization precision. The determined influence of asymmetric effects after 1,000 synchronization steps is depicted in Fig. 3(b). The y-axis shows a growing standard deviation of the time offsets between nodes with an about linear slope. However, though the precision degenerates, even in highly asymmetric scenarios, where more than 60% of the links are affected, a synchronization is possible.

**3. Accordance of internal clocks and external time:** Reaching external synchronicity can be realized by locally synchronizing some of the nodes with external time sources, e.g., authenticated signals of the Global Positioning System (GPS) [28]. However, to initially set up synchronization, the existing nodes must be set by an administrator to an approximate time window. Otherwise, the externally synchronized nodes will be detected as attackers. After that procedure, the externally synchronized nodes prevent the whole network from any potential long-term drift, as long as no more than 50% of the nodes suffer from a high, consistent drift.

**4. Integrity:** Due to the use of the VPN protection measures, external attackers are only able to drop or delay packets in the transport network. If this results in long-term asymmetric delays it may influence the algorithm up to a minor degree. For example, if a maximum RTT of one second is allowed, attackers may generate a maximum time failure of one second minus the real RTT when they are able to delay all packets of a certain node. Thus, an external attacker can only generate a limited bias at that node. A better synchronization in the presence of a long-term asymmetric delay is impossible.
In contrast to external attackers, internal ones can use compromised nodes to distribute arbitrarily forged timestamps. However, the simple introduction of a large offset cannot be expected to quickly lead to problematic situations. Nonetheless, a more intelligent attacker could emulate a consistent drift in order to "pull" uncompromised nodes away from the rest. In the context of network coordinate systems, such an attack is called *frog boiling* [29].
In order to evaluate the influence of such attacks several experiments were conducted. Herein, a number of attackers (usually 10% of the nodes) consistently increased their reported timestamps for 500 synchronization steps. Finally, the introduced standard deviation from the real time was measured and plotted.
Fig. 4(a) shows the impact of different attack drifts. Interestingly, the impact of the attack increases at first about linearly with the slope of the drift. However, for values of 25 $\frac{ms}{s}$ (and above) the attacker is not able to desynchronize the nodes at all, as its nodes are always filtered by the median. Thus, the internal attacker must estimate and emulate a suited slope to do optimal damage. Due to the hard cut, we assume the attacker to introduce a slope of 15 $\frac{ms}{s}$ in the following.
As the efficiency of the attack heavily depends on the number of compromised nodes, the ratio was varied for Fig. 4(b). Constituting these expectations the mean value grows about linearly with the ratio. However, the max-whisker increases much faster, indicating that in some situations more nodes are

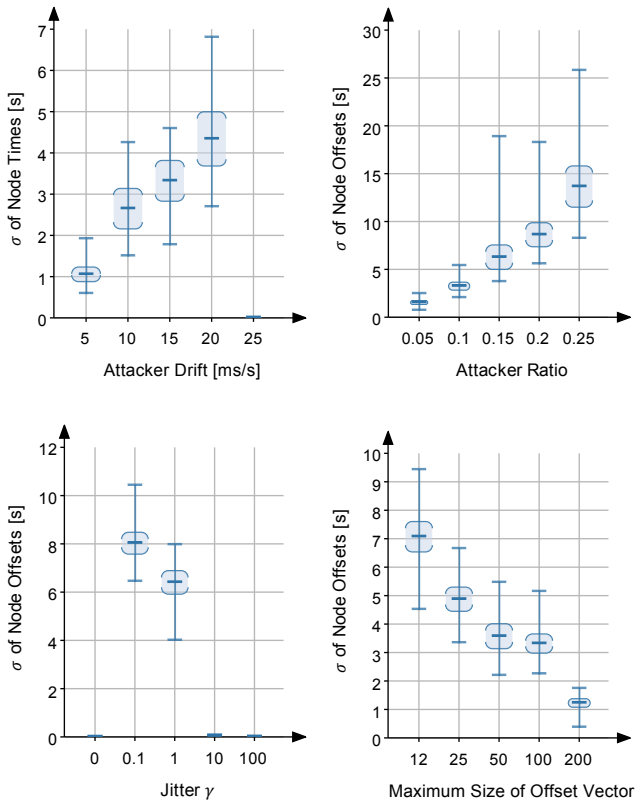



Fig. 4. Influence of internal attackers based of drift (a, upper left), ratio of compromised nodes (b, upper right), jitter (c, lower left), and queue length (d, lower right)

convinced by the attackers and eventually leading to snowball effects. Nonetheless, for VPNs even 10% compromised nodes can be considered a worst case scenario.

Fig. 4(c) depicts the results of an interesting scenario: the dependence of a successful attack on the jitter. Surprisingly, for very low and very high jitter the attack fails completely. The reason is two-fold: if very accurate measurements are possible, the attacker will very quickly ruled out by the median. If only very noisy measurements are possible, the attack takes more time as the synchronizing nodes only get few consistent values. Thus, attackers must adapt the drift to the jitter in the network and in both cases much lower values need to be realized.

As already outlined, the targeted agility of the synchronization process depends directly on the `maxOffsets` parameter. If adjustments are expected to occur seldom, large vectors will also assure that the impact of attackers is further limited. This correlation was also confirmed experimentally, as depicted in Fig. 4(d). Larger values efficiently limit the attacker, and due to the Zipf distributed deletion strategy even longer periods will be considered. The slightly higher value for 100 entries can be explained by optimization of the attack drift to that parameter.

**5. DoS resistance:** As the presented algorithm is fully distributed and localized, there is no single point of failure, whose compromise can have an influence on the synchronization. In order to desynchronize the VPN, attackers would have to prevent the communication within the whole VPN for a long time. The introduction of highly asymmetric delays can be considered more problematic, as nodes that are directly affected by a bandwidth exhausting attack detect a drift of the rest of the VPN. Without the assumption of additional unaffected uplinks, the only way to mitigate such attacks is the local disregard of outliers that are caused by such attacks. Within the presented algorithm, the offset vectors in combination with the Zipf distributed cleanup strategy assure that DoS attacks have to occur for a long time before leading to an effect on the synchronization. Longer DoS attacks, e.g., over days and weeks, seem unlikely with the current alertness of Internet providers. However, if such a threat should become presumable, more advanced detection schemes could be developed, e.g., by detecting measurable side effects like high jitter and stop synchronization during the attack.

**6. Robustness:** Due to the use of retransmissions and the RME, high packet error rates and jitter do not affect the synchronization of the VPN, as long as less than 50% packets are concerned. In case of a network partitioning, the divided subnets continue to synchronize, so that synchronicity within the partitions is always ensured. If the separation continues for a long time, the subnets can drift away from each other if there are differences in the averages clock drifts. As outlined in subsection IV-D, a later reunion may lead to situations where a synchronization is not achieved until the VPN overlay is connected in a way that no more densely connected node sets exist. Nevertheless, a synchronization will always be achieved again, if the overlay topology targets good expansion properties [30].

**7. Efficiency & Scalability:** As the synchronization messages are piggybacked in the heartbeats of the VPN overlay, and the values of nonce as well as timestamp are only 128 bits, the introduced communication overhead is minimal. The computing overhead is $\mathcal{O}(nm^2 \log m)$ for $n$ neighbors and queue length $m$, which poses no problem for current processors and reasonable sizes of $n$ and $m$ (i.e., $n = 15$ and $m = 100$).

Due to the logarithmic diameter of the VPN topology and the associated fast spread of timing information, the synchronization speed can be expected to grow only slowly with the number of nodes. The outcome of an experiment that was conducted to verify this assumption is shown in Fig. 5(a). Even though the x-axis is scaled logarithmically, the number of steps that were required to synchronize below an error of $0.1s$ grows only sublinearly. Furthermore, as indicated in Fig. 5(b), the synchronization precision that is achieved after 1,000 steps degrades only slowly when the number of nodes is increased from 50 to 400. In fact, since confidence intervals overlap no statistical significant effect can be determined. However, the magnitude of the values suggest that in undisturbed networks the precision solely depends on `minAdjOffset` and the network diameter. The paradoxically slightly worse precision in small VPNs can be explained by the larger effect of asymmetric paths in small overlays.

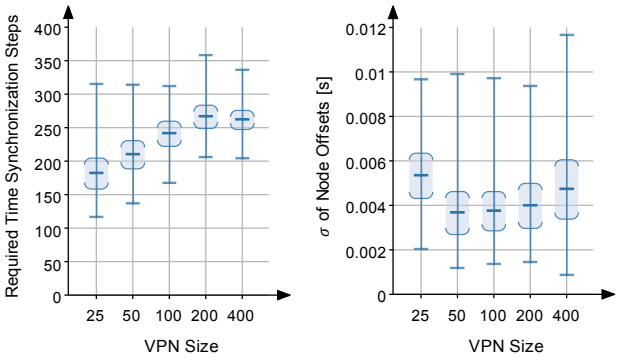



Fig. 5. Number of required synchronization steps (left) and achieved precision (right) over the number of nodes in the overlay

## VI. Conclusion and Future Work

In this article, a novel approach for distributed time synchronization in IP-based overlay networks and large VPNs was presented. The combination of robust estimation of time offsets and usage of the weighted median make it suitable to be used even if colluding internal attackers are present. Additionally, the proposed algorithm does not rely on exposed nodes, e.g., NTP servers or reference nodes, which could pose suitable targets for attackers. A temporary partitioning of the network is tolerable and occurs without a loss of inner synchronization in the partitions.

To evaluate the robustness against external and internal attackers, a simulation study was conducted and discussed. A powerful adaptive internal attacker that is able to manipulate timestamps, was shown to gain only little influence on the algorithm. If an adaptive attack occurs in many nodes, the achievable damage within the usable time frame is efficiently limited.

However, some issues remain to be improved in further studies. For example, it may be possible to adjust the weight of the confidence factors in order to gain a better trade-off between synchronization speed and attack resistance. Furthermore, it might be reasonable to blacklist nodes that generate outliers too often, and thus further reduce the abilities of a frog boiling attacker.

## References


[1] M. Rossberg, G. Schaefer, and T. Strufe, "Distributed Automatic Configuration of Complex IPsec-Infrastructures," *Journal of Network and Systems Management*, vol. 18, no. 3, pp. 300–326, May 2010.

[2] D. Mills, "RFC1305: Network Time Protocol (Version 3) specification, implementation and analysis," RFC 1305, Tech. Rep., 1992.

[3] S. Meiling, D. Charousset, T. C. Schmidt, and M. Wahlisch, "System-assisted service evolution for a future Internet — The HMcast approach to pervasive multicast," in *2010 IEEE Globecom Workshops*. IEEE, Dec. 2010, pp. 913–917.

[4] O. Gurewitz, I. Cidon, and M. Sidi, "Network Time Synchronization Using Clock Offset Optimization," in *IEEE International Conference on Network Protocols*. IEEE Computer Society, 2003.

[5] S. Samll, A. Terzis, F. Monrose, B. Doshi, and A. De Simone, "Scalable VPNs for the global information grid," in *Military Communications Conference, 2005. MILCOM 2005. IEEE*. IEEE, 2006, pp. 305–311.

[6] VeriSign, "The Trends And Changing Landscape Of DDoS Threats And Protection," VeriSIgn, Inc., Tech. Rep., 2009.

[7] D. Mills, "RFC4330: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI," pp. 1–28, 2006.

[8] R. Gusella and S. Zatti, "The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, 1989.

[9] F. Cristian, "Probabilistic clock synchronization," *Distributed Computing*, vol. 3, no. 3, pp. 146–158, Sep. 1989.

[10] O. Gurewitz, I. Cidon, and M. Sidi, "One-way delay estimation using network-wide measurements," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2710–2724, 2006.

[11] H. Karl, T. Lentsch, and H. Ritter, "Drahtlose Sensornetze," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 28, no. 2, pp. 66–67, Jun. 2005.

[12] K. Römer, P. Blum, and L. Meier, "Time Synchronization and Calibration in Wireless Sensor Networks," *Handbook of Sensor Networks: Algorithms and Architectures*, pp. 199–237, 2005.

[13] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, Jul. 2004.

[14] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[15] L. Ma, H. Zhu, G. Nallamothu, B. Ryu, and Z. Zhang, "Impact of linear regression on time synchronization accuracy and energy consumption for Wireless Sensor Networks," in *IEEE Military Communications Conference (MILCOM)*, 2008.

[16] S. Ganeriwal, S. Čapkun, C.-C. Han, and M. B. Srivastava, "Secure Time Synchronization Service for Sensor Networks," in *Proceedings of the 4th ACM workshop on Wireless security*. ACM, 2005, pp. 97–106.

[17] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 112–125, 2007.

[18] Y. Yang and Y. Sun, "Securing Time-synchronization Protocols in Sensor Networks: Attack Detection and Self-healing," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2008.

[19] K. Sun, P. Ning, and C. Wang, "Secure and Resilient Clock Synchronization in Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 24, no. 2, pp. 395–408, 2006.

[20] J. Ridoux and D. Veitch, "Principles of Robust Timing over the Internet," *Communications of the ACM*, vol. 53, no. 5, p. 54, 2010.

[21] L. Rizo, D. Torres, J. Dehesa, and D. Munoz, "Cauchy Distribution for Jitter in IP Networks," in *18th International Conference on Electronics, Communications and Computers*, 2008, pp. 35–40.

[22] A. F. Siegel, "Robust regression using repeated medians," *Biometrika*, vol. 69, pp. 242–244, 1982.

[23] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[24] J. R. Douceur, "The Sybil Attack," in *Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002, vol. 2429, pp. 251–260.

[25] A. Singh, T.-W. J. Ngan, P. Druschel, and D. S. Wallach, "Eclipse Attacks on Overlay Networks: Threats and Defenses," in *Proceedings of the 25th Conference of the IEEE Computer and Communications (INFOCOM)*, 2006.

[26] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *Proceedings of the European Simulation Multiconference (ESM'2001)*, 2001, pp. 319–324.

[27] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A Lightweight Network Location Service without Virtual Coordinates," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2005, pp. 85–96.

[28] J. Nielson, J. Keefer, and B. McCullough, "SAASM: Rockwell Collins' next generation GPS receiver design," in *IEEE Position Location and Navigation Symposium*, 2000, pp. 98–105.

[29] E. Chan-tin, D. Feldman, N. Hopper, and Y. Kim, "The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinate Systems," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 19, ch. 1.

[30] C. Gkantsidis, M. Mihail, and A. Saberi, "Random Walks in Peer-to-Peer Networks," in *Proceedings of the 23rd Conference of the IEEE Computer and Communications (INFOCOM)*, 2004.