PROCEEDINGS

# 55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

# Crossing Borders within the ABC

## Automation,

## Biomedical Engineering and

## Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

## Impressum
## Published by

Publisher: Rector of the Ilmenau University of Technology
Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff

Editor: Marketing Department (Phone: +49 3677 69-2520)
Andrea Schneider (conferences@tu-ilmenau.de)

Faculty of Computer Science and Automation
(Phone: +49 3677 69-2860)
Univ.-Prof. Dr.-Ing. habil. Jens Haueisen

Editorial Deadline: 20. August 2010

Implementation: Ilmenau University of Technology
Felix Böckelmann
Philipp Schmidt

## USB-Flash-Version.

Publishing House: Verlag ISLE, Betriebsstätte des ISLE e.V.
Werner-von-Siemens-Str. 16
98693 Ilmenau

Production: CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts

Order trough: Marketing Department (+49 3677 69-2520)
Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

## Online-Version:

Publisher: Universitätsbibliothek Ilmenau
ilmedia
Postfach 10 05 65
98684 Ilmenau

**Home / Index:**
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

# SCRUBBING THE VIVALDI NETWORK COORDINATE SYSTEM

*Franz Girlich*[1], *Michael Rossberg*[1], *Guenter Schaefer*[1], *Thomas Boehme*[2], *Jens Schreyer*[2]

[1] Telematics and Computer Networks Group [2] Discrete Mathematics and Algebra Group
Ilmenau University of Technology, Germany

## ABSTRACT

Over the last years network coordinate systems have gained much attention as they allow for an elegant estimation of distances between peer-to-peer endsystems. The most prominent representative of these approaches is Vivaldi, which is using a mass-spring-damper system to embed peers into a two-dimensional Euclidean coordinate space with an additional height coordinate to model access delays to the core network. In unimpaired overlay networks this simple method leads to a good approximation of pairwise delays.

Unfortunately, like most distributed algorithms, Vivaldi is likely to suffer from byzantine failures and several attack methods and countermeasures have been proposed. In this article we analyze theoretic bounds for a protection of Vivaldi network coordinates and show in particular how violations of the triangle inequality can in theory and simulation be exploited to create arbitrary movements.

***Index Terms***— Security, Peer-to-Peer, Network Coordinates, Vivaldi

## 1. INTRODUCTION

Distributed algorithms in the form of peer-to-peer overlay networks have gained much attention over the last years [1, 2, 3]. One of the often addressed problems is the optimization of overlay routing algorithms to underlay network topologies. In a simple case the distance within the overlay is not only minimized in each step, instead a greedy forward is performed by choosing the smallest quotient of overlay and underlay distance.

More complex systems adapt their overlay topologies to given underlay constraints. For example, some application layer multicast (ALM) systems create data distribution trees that are location aware [4]. The authors of RON [5] proposed an overlay network that exploits violations of the triangle inequality of the delay between its nodes. Thus, if it is quicker to forward data to an overlay node via a different overlay node than over a direct communication path the proxy is used. In order to determine the delays RON uses periodic pairwise measurements of the corresponding round-trip-times. The problem of this approach is its bad scalability as the number of transmitted packet grows quadratically to the number of overlay hosts.

To address this issue [6] proposed a decentralized approach to establish *network coordinate systems*. Subsequently, with the presentation of Vivaldi [7], a fully distributed algorithm became available to calculate artificial coordinates on basis of a spring-mass-model. As sketched in Fig. 1, a peculiarity of Vivaldi is the utilization of both a Euclidean and a Manhattan distance to model the access delay to the core network with a height coordinate, which reminds of an upside-down *brush*. After multiple rounds of calculations, the algorithm usually stabilizes, and the distances between coordinates indicate the actual delay between peers. Peer-to-peer mechanisms can use the coordinates to estimate the gain of topology changes before actually performing them. In particular it is possible for a node to forecast the round-trip-times between two other nodes.

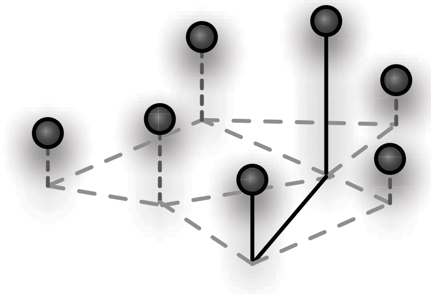

**Fig. 1**. Network coordinate space in Vivaldi: combination of Euclidean and Manhattan distance forming a *brush*

Unfortunately, Vivaldi has other drawbacks, which is most particularly its security. Without further protection mechanisms, a malicious node can claim arbitrary positions and delay probes. Kaafar et al. [8] have shown in various simulations, that by performing these attacks not only the own coordinates are influenced, but due to the spring-mass-model neighboring nodes are affected, too. Hence, it was only a consequence, that different countermeasures and counterattacks have been developed.

However, as none of the countermeasures has been successful for all attack models, the question arises whether comprehensive protection is possible at all. Thus, in this article we will perform the following contributions:

- the current state of research in Vivaldi protection is given,
- a mechanism to limit the influence of artificial delays is presented,
- several best practices towards a secure Vivaldi operation are derived, color=blue!30]Eigentlich machen wir es doch nur kaputt...
- and we perform a theoretical analysis of Vivaldi's interaction with the application layer, leading to a novel, potential security issue, which cannot be coped with under the assumption of a generic application overlay.

The rest of the article is structured as follows: First a background on the Vivaldi algorithm is provided, followed by an attacker model in section 3, which has to be considered by protection mechanisms. A state of the art analysis on the protection of network coordinate systems is presented in section 4. The main analysis of upper bounds for protection mechanisms is performed in section 5, followed by a short evaluation. The article closes with a conclusion and some directions for further work.

## 2. BACKGROUND

As the estimation of pairwise delays between the network hosts gained attention, virtual network coordinate systems moved into the focus. These systems basically map the nodes into a multidimensional coordinate space, where coordinates are calculated with respect to probed delays between the corresponding nodes. At the end of the computation, the distance between the coordinates of two nodes give an estimation of their network delay. A naive way to generate a coordinate approximation is to measure all pairwise delays first, move them to a designated node, and use a central optimization algorithm. This method obviously suffers from a poor scalability, much communication overhead, and a single point of failure. Hence, a system is needed that provides results of similar quality, but requires fewer samples and no central coordinator.

Many methods have been proposed to address this problem: Systems like Global Network Positioning (GNP) [6] or Practical Internet Coordinates for Distance Estimation (PIC) [9] are using fixed nodes with predefined coordinates called landmarks. Thus new nodes are able to compute their own coordinates based on measurements to a few landmark nodes. While these methods provide good results, they require designated nodes, whose compromise or failure may have dramatic effects on the whole system. As a result Dabek et al. proposed the fully distributed Vivaldi algorithm [7] that uses a method inspired by a mass-spring-damper system. Connections between nodes are illustrated to be springs, where the length of each spring is equivalent to the delay between the attached nodes. After a random initialization of the coordinates, the occurring forces move nodes until a stable position is reached. If the coordinate's distance of two nodes is smaller than length represented by the measured delay, the virtual spring is compressed, thus evokes a force pushing the nodes apart. Nodes that have too distant coordinates are pulled together in the same way. Using the physical model it is possible to show that all nodes coordinates converge to fixed points in finite time.

In order to use this principle for coordinate computation in a distributed fashion some further simplifications and modifications are necessary. To limit the communication and coordination overhead each Vivaldi node is only connected to a fixed number of neighbors, which are periodically asked for their coordinates and an error approximation. During the periodic polling the round-trip-times are measured and used to calculate a new iteration of algorithm 1. Its basic idea is to calculate the estimation error $e$ and move the nodes coordinate a tiny step towards the peers coordinate or away from it respectively. The absolute change in the coordinate is determined by the absolute error $e$ and the damping factor $\delta$ which itself depends on the error estimations $e_i$ and $e_j$. Each node maintains such an error estimation that expresses its confidence in its own coordinate.

**Algorithm 1** Decentralized Vivaldi using a dynamic damping factor

```
Require: Measured delay $rtt$, nodes coordinate $a_i$ peer coordinate $a_j$, estimated nodes coordinate error $e_i$, estimated peer coordinate error $e_j$, constant parameters $c_e$, $c_c$
  $w = \frac{e_i}{e_i+e_j}$                                   # Error balance
  $e = rtt - ||a_i - a_j||$                         # Absolute estimation error
  $u = \frac{a_i-a_j}{||a_i-a_j||}$                          # Direction of the movement
  $e_i = \frac{|e|}{rtt} c_e w + e_i(1 - c_e w)$     # New error estimation
  $\delta = c_c w$                                    # Dynamic damping factor
  $a_i = a_i + \delta e \times u$                           # Coordinate update
```

A variety of metrics can be used to embed the nodes into the virtual coordinate space. Dabek et al. [7] experimented with multidimensional Euclidean spaces, spherical coordinates, and their own development, a 2-dimensional Euclidean metric mixed with the manhattan distance, reminding of an upside-down brush and therefore called *brush space* in the following. Using the brush space, each node is mapped into a 2-dimensional space, that is augmented with a *height* as shown in Fig. 1. The distance between two neighboring nodes

$a = (a_1, a_2, a_h)$ and $b = (b_1, b_2, b_h)$ is then computed as $|b - a| = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2} + a_h + b_h$. As this metric allows an embedding of typical internet nodes with a low average error it is considered to be most suitable for Vivaldi.

## 3. ATTACKER MODEL & SECURITY OBJECTIVES

Overlay nodes executing Vivaldi can in theory be attacked by external as well as internal offenders, whereas a protection against externals can be achieved by utilizing encryption mechanisms like Datagram Transport Layer Security (DTLS) in combination with filters against delay attacks. More complicated is the possible influence of internal attackers, which may be able to influence:

- the coordinates that are reported to other nodes,
- the remote error value,
- round-trip-times measured by other nodes to the offender,
- and the overlay links chosen for measurement.

As internal attackers are realistic for most overlay networks and more powerful, we focus on them in this article.
Under these harsh conditions a perfect network coordinate system should give the following guarantees:

- **Convergence:** The network coordinates should tend to reflect the actual delays between hosts, and the calculated node distances should not change in a stable environment.
- **Immunity to selective moving:** Even with a low number of internal attackers, it should not be possible for them to move an arbitrary node too much in a certain direction.
- **Immunity to byzantine coordinates:** Attackers should not be able to lie themselves to self-chosen coordinates, e.g., in order to attract overlay traffic.

Depending on the upper layer application, and the type and number of attackers in the overlay network, these goals can only be achieved to a certain degree.

## 4. RELATED WORK

Although the Vivaldi algorithm provides a good estimation of the inter-node delays, it contains no protection against malicious nodes which delay their replies or even send arbitrary coordinates instead of their own. Kaafar et al. [10] developed several attacks based on these two methods that dramatically decrease the accuracy of the Vivaldi coordinates. As a result many mechanisms have been proposed to prevent malicious behavior or at least detect it:
**Statistical Filters:** Zage et al. [11] proposed an outlier detection mechanism that is based on temporal and spacial observations of the neighboring nodes behavior. In the embedding process a nodes coordinate is assumed to converge, thus causing the local error and the coordinates movement to drop. This characteristic is used by the temporal outlier detection to identify nodes, that "jump" through the coordinate space. The spacial detection is based on the assumption that a dependency exists between the coordinate's movement of the neighbors. Nodes, whose local error movement differs significant from the behavior of the other nodes are considered malicious and not used for the Vivaldi-algorithm. This method also prevents well embedded nodes from reducing their accuracy by communicating with newly joined nodes. Both mechanisms in combination can be used to reduce the set of capabilities an attacker has to reduce the systems accuracy, since his actions must not cause an exceedance of the preconfigured thresholds.
Kaafar et al. [12] consider the use of a Kalman filter [13] to achieve the same goal. Trusted observer nodes are set up whose neighbors are only other observers, thus these nodes can calibrate the filter's parameters so that the filter can predict the nodes movement through the coordinate space. Every normal node has to contact a close observer to obtain its parameters and run a filter by itself. If the update caused changes in the coordinate would differ too much from the predicted movement the update is rejected.
Both mechanisms can be used to reduce the attacker's space of action, however they are useless if applied alone, since Chan-Tin et al. proposed a counter attack [14].
**Veracity:** The basic idea behind Veracity [15] is to assign a set of verification nodes to each node that check the node's coordinates. After a node (the *initiator*) receives a reply from a neighbor (the *suspect*) it contacts the suspect's verification nodes to retrieve their vote. Each verification node compares the measured delay towards the suspect with the estimated distance determined by the Vivaldi-system. If the approximation error exceeds a threshold the verification node casts a negative vote. After the initiator collected all votes it follows the majority's decision and discards the suspects coordinate or passes it to the Vivaldi-algorithm.
While this method is a promising approach, it suffers from some practical issues such as bootstrapping and delay attacks against the verification nodes.
By now, no mechanism has been proposed that secures Vivaldi entirely and the question arises, if that is even possible. Hence, theoretical bounds for attack countermeasures are presented in the following section, that will reveal some of Vivaldi's inherent security problems.

## 5. THEORETICAL BOUNDS FOR ATTACK COUNTERMEASURES

Despite the outlined powerful internal attacker, in the following the overlay network is assumed to be protected by a hypothetical mechanism that perfectly prevents byzantine attacks as it is our intention to explore the theoretical bounds of countermeasures against Vivaldi attacks. In particular even each hostile node is only assumed to be able to choose the nodes it connects to, and it may delay packets for an arbitrary, though reasonable time. Due to the protection mechanism, attackers cannot lie about their own coordinates, estimated errors, and cannot answer quicker than their real delay.

### 5.1. Estimating upper bounds for delays

In order to further improve Vivaldi's resistance against delay attacks we propose a novel heuristic, based on *traceroutes* and the assumption that an attacker cannot compromise the core network. As the height of a Vivaldi coordinate models the access delay to a core network, every overlay host is assumed to initialize its own height with the delay to its first Internet router. And even though the height may fluctuate a little over time, it should always reflect the delay to a core router.
Exploiting this situation, a host may verify the position of another by performing a traceroute and finding the last core router of the path. To protect against malicious traceroutes answers, the following mechanisms are deployed:

1. Nonces in probe packets are used to prevent too early answers by attackers, possibly spoofing the address of a core router.
2. By querying databases, such as RIPE, it is verified that the last router belonging to the a different organization will be referenced.
3. Instead of utilizing the traceroute itself to measure delays, additional Internet Control Message Protocol (ICMP) echo requests are used to measure the delay to the last core router as well as the corresponding overlay node. This is because some router firmwares were found to delay answers to traceroutes, and thus caused anomalies.
4. Last-hop Routers and overlay nodes that discard tracesroutes are excluded from measurements and not used for reference.
5. Finally, in order to verify that a potential attacker did not add artificial peers to the path, an additional traceroute is performed targeted at the core router (see Fig. 2). If the intermediate nodes do not match, the overlay node either added peers, or (in rare cases) policy routing chose another path. In either situation the measurement is discarded.

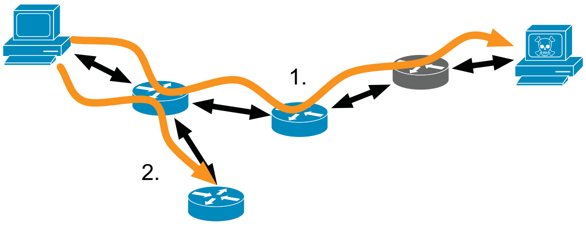

**Fig. 2**. Discovery of maliciously extended path by a second traceroute to last claimed core router

The obtained delay information is then used to check if $height \approx d_{node} - d_{router}$ holds, and if not the adjustment is discarded. Thus, when performing delay attacks the height must also be adjusted in order to generate valid updates. However, this will only cause the attacker to move into a maverick position. All in all, the heuristic allows for a feasibility check of reported delays up to a quality that solely depends on the allowed violation of the height model. It is expected that violations can be bound to $20ms$ for real life overlay networks, such the Planetlab.

### 5.2. Using asymmetric links

In contrast to a real life spring-mass-model, the authors of Vivaldi did not specify that the neighborhood relation must be symmetric. Thus, if a node queries another node for a position update and latency measurement, the other node does not have to take the quering node into account. From a security perspective this asymmetry is a desired property as it prevents malicious nodes to perform directed attacks.
For example, consider an attacker who wants to move a particular node to a different coordinate, e.g., to influence the routing. If links were symmetric, he only needs to put his own nodes physically into a position, where the triangle inequality is violated towards the attacked node, and let his nodes connect. The attacked node would now consider its own position to be wrong, and move in the controlled direction. In order to prevent such an attack, nodes must either be free to chose their neighborhood free of symmetry constraints, or the legitimacy of neighborhood relation must be proven by the upper layer overlay algorithm. As this would tremendously reduce the applicable deployment scenarios for Vivaldi, asymmetric links will be assumed for the rest of the article.

### 5.3. Aggregation of measurement steps

While using Vivaldi as described in section 2 convergency problems arise if nodes violate the triangle inequality as shown in Fig. 3. In this example network node $b$ is asymmetrically connected to the nodes $a$ and $c$ while these however will not take an active part in the

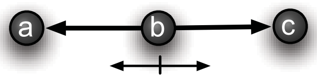

**Fig. 3**. Three nodes that violate the triangle inequality.

system. After requesting the coordinates from $a$ and $c$, $b$ will move towards the first node that answers, thus decreasing the estimation error with this node while increasing the error with the other. When the second reply arrives, $b$ will move towards the other node in the same way. Even if all nodes are initially optimally embedded, node $b$ will still oscillate around its optimal position without ever reaching it again. If there were directed circles in the neighborhood graph, the network might even move through the coordinate space.

To solve this problem, the algorithm has to compute the change in the coordinates for each arriving reply and aggregate them instead of applying them directly. This way oscillations can be reduced and the example above will converge immediately, thus aggregated updates will be assumed in the following.

### 5.4. The simple crawler anomaly

However, due to the allowance of asymmetric links, Vivaldi does not exactly behave like a spring-mass-model anymore and certain anomalies occur. Consider the very simple network shown in Fig. 4, consisting of three honest nodes that violate the triangle inequality. In the constructed situation the nodes $a$ and $b$ as well as $b$ and $c$ are too far apart while $a$ and $c$ are too close together. Despite the network's simple structure its coordinates will never converge, but move with a constant speed in one direction. Due to the inducted movement pattern, we named this effect a *crawler anomaly*.

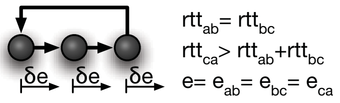

**Fig. 4**. Crawler anomaly: three honest nodes with asymmetric neighbor relations. The arrows are drawn from the initiator to the neighbor.

We assume that any pair of nodes $i, j$ is initially optimally embedded into an multidimensional Euclidean space, with respect to the sum of the squared errors $e_{ij} = | \, ||i - j|| - rtt_{ij} \, |$. In this case the optimal coordinates are located in one line as shown in Fig. 4 with $e_{ab} = e_{bc} = e_{ac}$. To simplify the upcoming calculations, $\hat{\delta}$ is defined to be the lower bound of the dynamic damping factor $\delta$. Since $\delta$ depends on the measured coordinate errors, which cannot converge to zero due to the triangle inequality violation, such a lower bound exist and is greater zero. Since nodes' coordinates are already arranged in a line, the coordinates can furthermore be considered to be one-dimensional.

Because $a$ only uses $b$'s coordinates to update its own, it will move a at least $\hat{\delta}e_{ab} = \hat{\delta}e$ units towards $b$. The same goes for node $b$ when communicating with $c$. Since $c$ and $a$ are too close together, $c$ will move at least $\hat{\delta}e$ away from $a$ after obtaining its coordinates. Thus, all three nodes moved at least $\hat{\delta}e$ into the same direction without any changes to their distance, when assuming that all nodes update their coordinates simultaneously. Since $\hat{\delta}$ as well as $e$ are strictly greater than zero, the whole network will never stop its movement.

While this anomaly works as described above with the use of an Euclidean coordinate space, it will however behave differently if applied to Vivaldi's brush space. Here, nodes $a$ and $b$ (forming the *tail*) are also *pulled*. In difference to the Euclidean version, every step decreases the height component of the coordinates, thus the tail is attracted to the ground of the brush. Node $c$ (the *head*) measures a delay against $a$ that is too large and is *pushed* away from the tail. Additionally, the head's height component is constantly growing until it is equal to the delay to $rtt_{ac} - rtt_{ab}$. After that, $b$'s position is right underneath $c$, since this place is the closest to $c$, $a$ is located $rtt_{ab}$ away from $b$, and the network converges. Hence, modifications are necessary to achieve the same movement within a brush space.

### 5.5. The pegged crawler anomaly

In order to create a movement, a fourth node (the *peg*) is introduced, which maintains a symmetric neighbor relation to the crawler's head, as shown in Fig. 5. Because the delay between head and peg is much smaller than between the other nodes, the head's height is bound to a low position. Again, nodes are arranged in a line so that it suffices to consider a one dimensional affine subspace and an additional height. As in the previous section, $\hat{\delta}$ is assumed to be constant and considered to be a lower bound for the real dynamic $\delta$. We further assume, that nodes update their coordinates cyclically using the aggregated Vivaldi-algorithm as presented in section 5.3, and in the following order: $a \to b, b \to c, p \to c, c \to a, p$.

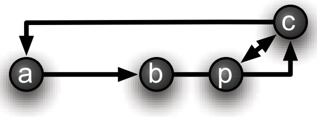

**Fig. 5**. Pegged crawler anomaly: simple crawler anomaly augmented with a fourth node $p$ that has a symmetric neighbor relation to $c$

**Lemma.** Let $B = \{(a, a_h), (b, b_h), (c, c_h), (p, p_h) \mid a, b, c \in \mathbb{R}, a_h, b_h, c_h \in \mathbb{R}^+\}$ be the state space of previously described network. Then there is a Vivaldi state

$s = \{(a, a_h), (b, b_h), (c, c_h), (p, p_h)\} \in B$, so that the next-iteration state $s' = \{(a', a'_h), (b', b'_h), (c', c'_h), (p', p'_h)\} \in B$ equals:

$$s' = \{(a+\Delta, a_h), (b+\Delta, b_h), (c+\Delta, c_h), (p+\Delta, p_h)\}$$

with $\Delta > 0$.

*Proof.* Let $d_{ij} = |i - j|$ be the distance of node $i$ and $j$ in the Euclidean dimension (heights not included). We now define $v$, $e$, $d_{ab}$, $d_{bc}$, $d_{ac}$, and $d_{pc}$ as the following:

$$\begin{aligned} v &= rtt_{ac} - (rtt_{ab} + rtt_{bc}) \\ e &= \tfrac{1}{5+2\hat{\delta}} v \\ d_{ij} &= e + rtt_{ij}, \text{ for } (i,j) \in \{(a,b), (b,c), (p,c)\} \\ d_{ac} &= d_{ab} + d_{bc}. \end{aligned}$$

Furthermore, we set $a_h = b_h = p_h = 0$ and $c_h = \epsilon$, with $\epsilon$ close to zero. For the nodes $a, b, p, c$ we get the following update equations:

$$\begin{aligned} a' &= a + \hat{\delta}(d_{ab} - rtt_{ab}) \\ b' &= b + \hat{\delta}(d_{bc} + \epsilon - rtt_{bc}) \\ p' &= a + \hat{\delta}(d_{pc} + \epsilon - rtt_{pc}) \\ c' &= c + \tfrac{\hat{\delta}}{2}(rtt_{ac} - d_{a'c} - \epsilon + rtt_{pc} - d_{p'c} - \epsilon). \end{aligned}$$

After the insertion of the defined values we get:

$$\begin{aligned} a' &= a + \hat{\delta}e \\ b' &= b + \hat{\delta}e + \hat{\delta}\epsilon \\ p' &= p + \hat{\delta}e + \hat{\delta}\epsilon \end{aligned}$$

and

$$\begin{aligned} c' &= c + \tfrac{\hat{\delta}}{2}(rtt_{ac} - d_{a'c} + rtt_{pc} - d_{p'c}) - \hat{\delta}\epsilon \\ &= c + \tfrac{\hat{\delta}}{2}(rtt_{ac} - d_{ac} + rtt_{pc} - d_{pc} - 2\hat{\delta}e - \hat{\delta}\epsilon) - \hat{\delta}\epsilon \\ &= c + \tfrac{\hat{\delta}}{2}(rtt_{ac} - d_{ac} - e - 2\hat{\delta}e - \hat{\delta}\epsilon) - \hat{\delta}\epsilon \\ &= c + \tfrac{\hat{\delta}}{2}(rtt_{ac} - d_{ab} - d_{bc} - e - 2\hat{\delta}e - \hat{\delta}\epsilon) - \hat{\delta}\epsilon \\ &= c + \tfrac{\hat{\delta}}{2}(v - (3 + 2\hat{\delta})e - \hat{\delta}\epsilon) - \hat{\delta}\epsilon \\ &= c + \tfrac{\hat{\delta}}{2}((5 + 2\hat{\delta})e - (3 + 2\hat{\delta})e - \hat{\delta}\epsilon) - \hat{\delta}\epsilon \\ &= c + \hat{\delta}e - \tfrac{\hat{\delta}^2\epsilon}{2} - \hat{\delta}\epsilon. \end{aligned}$$

We will leave out the calculation for $c'_h$ at this point, but it can be seen that $c'_h < c_h = \epsilon$, so the height of the head also tends to zero. If we set $\epsilon = 0$ and assume, that the nodes $a$, $b$ and $c$ violate the triangle inequality, the lemma's proposition is true with $\Delta = \hat{\delta}e > 0$. □

In order experimentally validate that the described attack can be successfully launched on the full Vivaldi algorithm, simulations utilizing OMNeT++ have been performed. Within the simulation the four nodes have been arranged like in Fig. 5. In contrast to the proof, all coordinates have been randomly initialized. Still, as shown in Fig. 6, the center of the network is exposed to a constant movement in all 64 runs. It also indicates that neither a fixed nor a specific communication order is necessary for the anomaly to work.

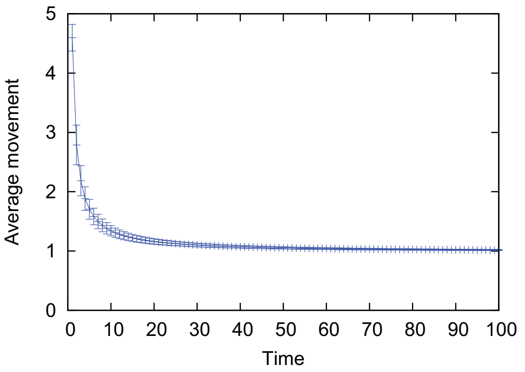

**Fig. 6**. Average movement of the crawler-network after random initialization and 95% confidence intervals of 64 runs

## 6. CONCLUSION & FUTURE WORK

The need for quickly adopting overlay networks and the related development of network coordinate systems, such as Vivaldi, have gained much influence recently. Despite the emergence of security measures for these distributed algorithms, no work has been done to show the limits of such work. This article tries to close the gap by identifying several issues and defining novel countermeasures. However, as we were able to show, Vivaldi has an built-in weakness concerning violated triangle inequalities. These allow attackers for a generation of valid network configurations that result in a movement of the overall overlay.

The focus of our future research is on the identification of further network motifs that create instabilities in Vivaldi. Using this knowledge, we will construct specially adapted overlay algorithms that are invulnerable to the sketched attacks, by avoiding instable configurations.

## 7. REFERENCES

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001, pp. 329–350.

[3] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed Hashing in a Small World," in *Proceedings of the 4th Symposium on Internet Technologies and Systems (USITS)*, 2003.

[4] T. Strufe, J. Wildhagen, and G. Schaefer, "Network-Efficient and Stable Overlay-Streaming Topologies (German: Netzwerkeffizienz und Stabilität von Overlay-Streaming-Topologien)," in *Bi-annual GI Symposium Communication in Distributed Systems (KiVS)*, 2007, pp. 3–14.

[5] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 131–145.

[6] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," in *Proceeedings of the 21st Conference of the IEEE Computer and Communications (INFOCOM)*, vol. 1, 2002, pp. 170–179.

[7] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2004, pp. 15–26.

[8] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, "Real attacks on virtual networks: Vivaldi out of tune," *SIGCOMM'06 Workshops*, Sep. 2006.

[9] M. Costa, M. Castro, A. Rowstron, and P. Key, "PIC: Practical Internet Coordinates for Distance Estimation," in *In International Conference on Distributed Systems*, 2003, pp. 178–187.

[10] M. A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, "Real attacks on virtual networks: Vivaldi out of tune," in *LSAD '06: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*. New York, NY, USA: ACM, 2006, pp. 139–146.

[11] D. J. Zage and C. Nita-Rotaru, "On the Accuracy of Decentralized Virtual Coordinate Systems in Adversarial Networks," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2007, pp. 214–224.

[12] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, and W. Dabbous, "Securing Internet Coordinate Embedding Systems," in *In Proceedings of the ACM SIGCOMM*, vol. 37, no. 4. New York, NY, USA: ACM, 2007, pp. 61–72.

[13] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME – Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[14] E. Chan-Tin, D. Feldman, Y. Kim, and N. Hopper, "The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinates," 2009.

[15] M. Sherr, B. T. Loo, and M. Blaze, "Veracity: A Fully Decentralized Service for Securing Network Coordinate Systems," in *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)*, Feb. 2008.