On Distributed Geolocation by Employing Spring-Mass Systems

Michael Grey and Michael Rossberg and Martin Backhaus and Guenter Schaefer
Ilmenau University of Technology
[michael.grey, michael.rossberg, m.backhaus, guenter.schaefer][at]tu-ilmenau.de

***Abstract*—Recently, finding the geographic whereabouts of nodes became a key service for many distributed applications, e.g., online games or localizing delivered content. However, an exact localization may be impossible because of GPS signals being unavailable, receivers too expensive, or energy too scarce. Hence,alternatives emerged that typically rely on central databases, which in turn are often found to be inaccurate, though. Facing that problem, we study a complementary idea: By constructing a delay-weighted spring-mass embedding of nodes and augmenting the system with geographic hints, e.g., those of traditional location databases, we efficiently estimate geographic positions of nodes by multilateration and solely distributed means. We will show that peer positions can be estimated with an accuracy of a few hundred kilometers in the average case. The proposed system is evaluated by simulations that are based on real-world PlanetLab latency data.**

***Keywords*—Geolocation, Spring-Mass System, Virtual Coordinates, Dynamic Anchors, Resilience, Overlay Networks.**

I. Introduction

With the emergence of the Internet, the development of efficient, globally distributed applications became possible and increasingly popular, for example, private overlay networks, online games, and sensing applications to name just a few. Nonetheless, the given examples have one thing in common: they sometimes require geolocation information. Overlay networks may plan backup routes based on geographic coordinates [19], matchmaking algorithms in online games depend on that information [1].

However, the exact location of nodes might be unobtainable, for Global Positioning System (GPS) signals being unavailable, receivers being too expensive, or energy too scarce. Even if GPS receivers are available, fuzzy location information may help to initialize the device more quickly, thus significantly reducing energy consumption [13]. Therefore, several other approaches emerged, e.g., localization by nearby WLAN BSSIDs and the mapping of IP addresses to geolocations. However, these approaches rely on central databases that are – if solely used – often extremely inaccurate [21], [17].

Hence, within this article we want to apply and study a different approach: By constructing a delay-weighted spring-mass embedding of nodes and augmenting the system with geographic hints, we focus on efficiently estimating geographic positions of nodes by multilateration and distributed means. In the course of the article, we point out the potentials and limits of this method in terms of achievable location accuracy. Naturally, by solely relying on delay measurements we cannot expect to achieve exact localization at global scale, i.e., at city level for every participant, but such a resolution is not required for all distributed applications, e.g., backup routing that addresses catastrophic infrastructure failures [15]. Furthermore, the proposed scheme provides location estimates by employing a method, which is complementary to mapping-based approaches. Thus, it may also be applied to overcome pitfalls of the latter approaches by detecting outliers.

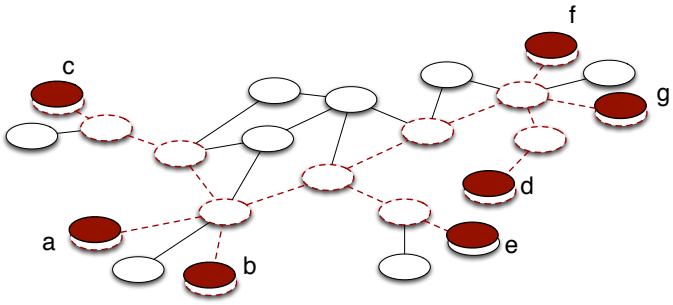


Fig. 1. Exemplary overlay on top of a transport network; as in real systems, overlay links typically often share physical connections.

The presented approach has its roots in the virtual network coordinate approach Vivaldi [5], however it provides several means to work accurately with geographic coordinates. To provide robust geolocation information for all members of an overlay or distributed application (as indicated in Fig. 1), we make the following contributions:

- we categorize existing distributed geolocation methods,
- survey major sources of inaccuracy of distributed geolocation by delay measurements,
- we derive enhancements to the spring-mass approach including involvement of intermediate systems and so-called *fuzzy anchors*,
- and evaluate the mechanism against data from PlanetLab measurements.

The remainder of this article is organized as follows: The next section will give objectives for distributed geolocation mechanisms, which are used to categorize the related work in Sec. III. Afterwards, a short overview on potential sources of inaccuracy of geolocation by virtual coordinates is presented in Sec. IV. Sec. V contains details on our approach, which is subsequently evaluated and discussed in Sec. VI. The article

SPONSORED BY THE
Federal Ministry of Education and Research

concludes in Sec. VII.

II. System Objectives

For a reliable, globally distributed estimation of whereabouts the following objectives should be met:

- **No sole dependence on external location information:** While for some sites it may be feasible to obtain locality information, i.e., by picking out hints from local hostnames or correlating location databases, an access to reliable information is not practically available for all nodes. Thus, a mechanism should make use of external locality references where possible, but treat this information with caution.
- **Accuracy:** The derivation of estimated and real position is the substantial metric for geolocation. For distributed geolocation without reliable external information we aim at the best precision that is possible by delay measurements, which can be expected to be a few 100 kilometers.
- **Scalability:** Practically relevant distributed systems can easily reach sizes of thousands of participants; hence, a mechanism must scale over the number of participants and depend on local knowledge only.
- **Robustness:** The distributed geolocation system should be insensitive to measurement failures and inaccuracies, and be robust in case of node and link failures. Thus, exposed entities are to be avoided.

The following will discuss existing approaches for geolocation with regard to these requirements.

III. Related Work

Several concepts were formerly presented to perform geolocation in distributed systems by assessing delay measurements. For example, the authors of *GeoPing* [16] propose to measure the delay to n designated anchors. The resulting values are then compared in an n-dimensional vector space to previously defined references. Finally, the position belonging to the reference vector with minimum Euclidean distance is adopted as the peer's position. *GeoPing* is centralized as the availability of all reference anchors is a prerequisite. Furthermore, the authors of [2] state that the prediction quality is solely valuable for small average delays and rapidly decreases for more significant delays, i.e., in global scenarios.

Similarly, *Constraint-based Geolocation (CBG)* [9] relies on initial delay measurements. Afterwards, linear regression is applied to the collected data to perform multilateration. However, all-in-all *CBG* is also centralized, making use of complex computations for host localization. Building on this procedure, the authors of [11] propose the so-called *Topology-based Geolocation*, which also takes intermediate systems into account. The provided optimization problem for localization includes geographic hints given by the DNS names of intermediates. While the involvement of intermediate systems seems to significantly increase the quality of the predictions, the use of DNS introduces third-party-system dependencies and the centralized optimization reduces robustness as well as scalability.

Various other approaches share such characteristics. There is a variety of hybrid approaches, e.g., the prominent *Ono* plugin [4], which relies on so-called content distribution network oracle. Furthermore, beyond the mainly measurement-based approaches mentioned so far, there are several concepts that focus on geolocation by database lookups only. This includes, among others, basic extensions to DNS records [6], *GeoCluster* [16], and *DNS-LOC* [14] as well as various proprietary solutions. The principle of these approaches is the same: databases or directories are queried with the public IP address of the looked-for peer. However, this principle generally suffers from dynamic address assignment and the subsequent invalidation of database records [7]. Furthermore, Shavitt et al. [21] and Poese et al. [17] recently studied the accuracy of database-querying approaches. Results indicate that the location accuracy is not generally reliable, even failures of thousands of kilometers were observed. Database approaches generally achieve good results for a major amount of hosts, though.

Besides geolocation, other means of efficient and scalable distance approximation exist: so-called network coordinate systems. The authors of [12] categorize network coordinate algorithms into two main classes: On the one hand, *Landmark-based schemes*, e.g., GNP and Lighthouses, typically rely on a fixed number of landmark peers to calculate their virtual coordinate. On the other hand, *Simulation-based schemes* calculate coordinates in a fully distributed way. Popular examples are the *Vivaldi* [5] approach that makes use of spring compression and relaxation, as well as *Big Bang Simulation* [20] that relies on the simulation of an explosion of particles under a force field. An interesting approach called *Htrae* [1] uses a spherical coordinate system and embeds nodes initially by their geographic coordinates to increase convergence speed. Even though Htrae bears resemblance to our approach at first glance, it is not designed for reflecting actual geographic coordinates and may for example treat the spherical coordinate embedding in a simpler way.

IV. Geolocation Based on Delay Data: Potential Pitfalls

A distributed geolocation based on delay measurements will most likely suffer from various inaccuracies, such as:

Dominating delay type: To allow for coordinate estimations a general correlation between network delays and geographic distance between hosts must be presumed. Thus, the minimum value of delay samples between peers must be dominated by propagation delays, rather than queueing or processing delays. In fact, at least on global scale delays are widely expected to be dominated by propagation delays [3], even though being more or less *disturbed*.

Network layer abstraction: The propagation delay between hosts depends on their network connectivity. However, already the physical connections are seldom beeline-like, but restricted by infrastructural constraints. As for a multilateration on the surface of a globe at least three different paths are required, and most nodes do not have three different uplinks, the achievable resolution is limited to the distance between the nodes and the first router that has at least three different uplinks. Furthermore, for policy reasons packet delivery cannot be expected to take shortest paths. Any of these effects leads to a degradation of the correlation between end-to-end distance and delay.

Asymmetric delays: While distributed measurements without exact clock synchronization can only use delay approximation by Round-Trip Time (RTT) measurements, real latencies are

often asymmetric depending on the direction [18], i.e., due to routing path asymmetries.

Due to these effects, delays between Internet peers sometimes violate the *triangle inequality* [12]. Formally, given the delay $d(A, B)$ between two nodes A and B a triangle inequality violation is the case if there is any other peer C that satisfies

$$d(A, B) > d(A, C) + d(C, B)$$

By degrading the correlation between distance and latency also the quality of embeddings suffers. All-in-all, a distributed geolocation by delay measurements cannot be expected to achieve highest accuracies. Nonetheless, entirely distributed geolocation approaches can be expected to achieve high significance, as path asymmetries can be filtered considerably better than in centralized systems, for example.

V. Distributed Geolocation by a Delay-based Node-Embedding

Virtual coordinate systems allow for a delay-based embedding of Internet nodes into a virtual vector space, such that the distance in the vector space correlates with network delays. As global latencies are dominated by propagation delay, one could expect that such an embedding would be best fitted into a spherical space, following the structure of the Earth, and therefore directly leading to coordinates that correlate with geographic coordinates. However, prior works found that a combination of a 2-dimensional Euclidean and an 1-dimensional Manhattan space better reflects the real world network delays [5]. Reason for this are several structural inhomogeneities in the Internet, which lead to this issue, e.g., the low number of direct links between Europe and East Asia.

Nonetheless, the 2-dimensional-Euclidean part of the coordinates can still be embedded into a spherical space. Furthermore, the described effects will tend to disappear as the Internet becomes more dense. Therefore, we will adapt the Vivaldi network coordinate system to work directly with geographic coordinates in the following. As will be shown in evaluation, the occurrence of additional deformations due to spherical spaces will be widely prevented.

Algorithm 1: Classic Vivaldi update algorithm: d denotes to the delay between peers, e_i labels the estimated error of the own peer coordinate, and e_j refers to the error of the neighbor coordinate. $\vec{x_i}$ reflects the virtual coordinate of peer i

$$w \leftarrow \frac{e_i}{e_i + e_j}$$
$$e_s \leftarrow \frac{|\|\vec{x_i} - \vec{x_j}\| - d|}{d}$$
$$e_i \leftarrow e_s \cdot c_e \cdot w + e_i\,(1 - c_e \cdot w)$$
$$\vec{\delta} \leftarrow c_c \cdot w \cdot (d - \|x_i - x_j\|) \cdot (\vec{x_i} - \vec{x_j})$$
$$\vec{x_i} \leftarrow \vec{x_i} + \vec{\delta}$$

A. Spring-Mass Based Embedding in Spherical Coordinates

As already mentioned, Vivaldi provides distributed means to calculate *virtual* coordinates. Algorithm 1 contains the basic rules for coordinate adaption in the classical Vivaldi approach. However, the algorithm serves for illustration purposes only, as we will rely on a numerically more stable version [8] in the following. Additionally, we deploy a Manhattan space component (the so-called *height*) for each coordinate, which is already known from Vivaldi and models the latency component a node experiences towards all neighbors, e.g., due to a significant Internet access delay.

To adopt the system to *real* coordinates, all calculations that were formerly performed in Euclidean space now have to be performed in spherical space. However, as an exact computation of adaptions in spherical space requires the computational intensive solution of differential equations, we decided to perform the following iterative procedure:

First, the spherical distance d as well as the azimuth α towards each neighbor is computed. Whereas, the azimuth α denotes the angle between the straight line through the neighbor's coordinates and an imaginary straight line leading from the node's coordinate towards the North Pole. According to a special case of the *Vincenty formula* [23], given the sphere radius r, the spherical distance d between two points on the surface defined by pairs of latitude and longitude (ϕ_1, λ_1) and (ϕ_2, λ_2) can be calculated in a *numerically stable* way as follows:

$$\Delta\lambda = \lambda_2 - \lambda_1 \tag{1}$$
$$\frac{\sin\sigma}{\cos\sigma} = \frac{\sqrt{(\cos\phi_2 \sin\Delta\lambda)^2 + (\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos\Delta\lambda)^2}}{\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\Delta\lambda}$$
$$d = r \arctan\left(\frac{\sin\sigma}{\cos\sigma}\right) \tag{2}$$
$$\alpha = \arctan\left(\frac{\sin(\Delta\lambda)\cos(\phi_2)}{\cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\Delta\lambda)}\right) \tag{3}$$

Using these values, all neighbors are projected on a tangential plane that originates at the coordinate representation of the node to be adapted. The method is comparable to a generalized *Gnomonic projection* that contains both hemispheres (see Fig. 2).

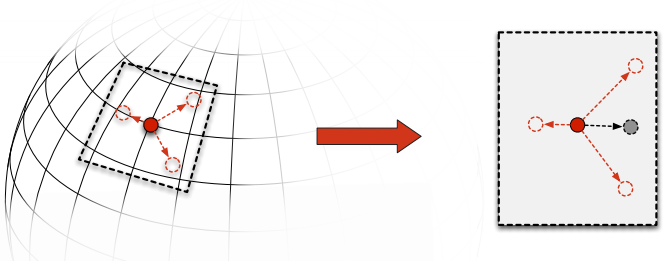


Fig. 2. Projection from virtual earth towards local tangential plane and cumulative adaption to three neighbors.

Second, an adaptation vector $\vec{\delta}$ is calculated for every neighbor using a spring-mass update algorithm.

Third, the final adaptation vector is determined by averaging all resulting vectors, which is then projected on the spherical space. The necessary angle α and length d are extracted from the tangential plane representation. Given a starting point (ϕ_s, λ_s) the resulting coordinates after adaption

(ϕ_r, λ_r) can be determined as follows (see Fig. 3):

$$\sigma = \frac{d}{r} \tag{4}$$

$$\phi_r = \arcsin\left(\sin\phi_s \cos\sigma + \cos\phi_s \sin\sigma \cos\alpha\right) \tag{5}$$

$$\Delta\lambda = \arctan\left(\frac{\sin\alpha \sin\sigma \cos\phi_s}{\cos\sigma - \sin\phi_s \sin\phi_r}\right) \tag{6}$$

$$\lambda_r = \lambda_s + \Delta\lambda \tag{7}$$

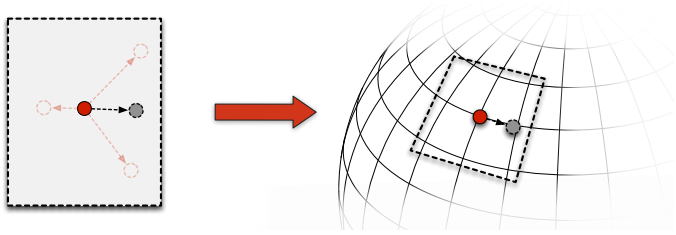


Fig. 3. Projection of resulting adaption vector from tangential plane to geographic coordinates.

It should be noted that the projection described above makes an inherent error, especially if the distances are large or the coordinates are close to the poles. However, this becomes negligible when the adaptation step is repeated, i.e., rerunning the adaptation step in only five iterative steps leads to an error within acceptable range.

B. Fuzzy Anchors

Using the outlined algorithm, nodes are able to estimate their relative geographic distance, but not absolute geographic coordinates. Thus, additional geographic hints must be regarded in the calculation process. Intuitively, this may happen already at coordinate initialization, i.e., nodes that know about their country or zip code area can select coordinates accordingly. However, by doing so nodes with a higher churn rate have more influence on the system. Furthermore, at low churn rates the system may begin to drift, like Vivaldi does, when coordinates are only considered at insertion time.

In order to adapt the system we suggest a broader approach: so-called *fuzzy anchors*. Traditional fixed anchors may suffer from inaccurate position information, are typically exposed points for failures, and may only regard information from one source, e.g., one geo database. However, in our approach the nodes with position information generate additional *virtual neighbors* for each of the hints' coordinates. The amount of attraction induced by the marks depends on the projected quality of the hint, which reflects the accuracy of position information (e.g., high accurate GPS information or comparably imprecise zip code areas).

While this attraction is involved in every local calculation, it is neither directly distributed nor taken into account by any other peer. The involvement of hints is illustrated in Fig. 4. Like in the original Vivaldi approach, all nodes select a defined constant number of neighbors, e.g., node a selected b, c, d and e. In addition, a is aware of an external position hint in terms of a fixed geographical coordinate denoted by m. The coordinate adaptation transparently processes the attractions of neighborhood nodes as well as the attractions by one or more fuzzy anchors. Accordingly, position hints locally participate

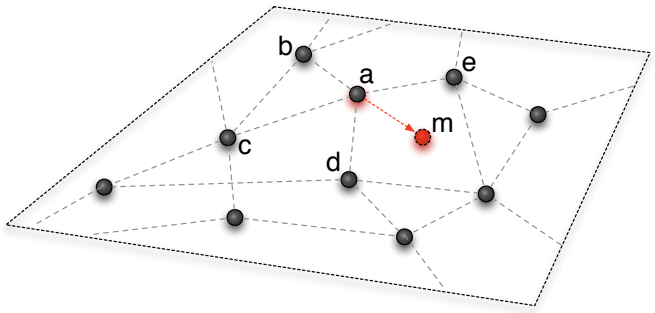


Fig. 4. Node a maintains a fuzzy anchor m, which attracts a to the hint location.

in every update procedure in the same way as other neighbor coordinates do. In contrast to the naive incorporation of fixed anchors, the update procedure may treat inaccurate or even erroneous external location information robustly.

C. Integration of Intermediate Systems

To reduce potential inaccuracies, we propose the integration of coordinates for intermediate systems. This is expected to further increase the prediction quality:

First, it enables the differentiation between geographically more distant and nearby hosts that only have a high access delay. Considering for example two nodes a and b that are connected by a slow DSL connection and a distant node c that is connected directly to the core network. When utilizing latency measurement only, all peers appear to belong to the same site. Involving network layer information, such as measured delays towards routers, allows to differentiate in such situations.

Second, given end-to-end path measurements, the deviation of network and line-of-sight path increases with every intermediate hop. Fig. 5 illustrates an example: The line-of-sight – and thus, the expected delay – between overlay nodes a and b is short in comparison with the network topology path traversed by packets. This is due to physical topology constraints, e.g., the non-existing submarine cables between Scandinavia and Britain. By additionally calculating coordinates for intermediate routers, these inaccuracies decrease. Furthermore, disturbances like substantial processing delays are reflected at the specific intermediate, which leads to an increased disturbance locality. Note that the intermediate nodes are not required to participate themselves actively, the necessary calculations can be distributed to the overlay nodes.

In order to identify intermediate systems we rely on periodic *traceroutes*. The obtained IP addresses of traversed routers are mapped to specific intermediate entities, where dealiasing and reduction techniques [10] are applied to aggregate equal systems with varying responding interfaces on paths towards different destinations. Furthermore, traceroute probes implicitly return the RTT for each detected intermediate. The RTT between intermediates on one dedicated path can thus be approximated by subtracting RTTs of consecutive intermediates. This methodology, however, may exhibit inaccuracies, e.g., due to asymmetric paths or varying processing delays at some intermediates. In order to mitigate some of the probing inaccuracies, multiple measurements are used for delay calculation:

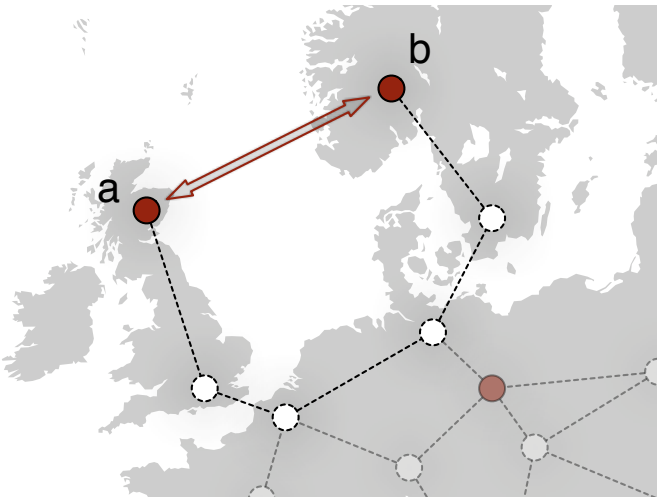


Fig. 5. Exemplary topology section with high discrepancy between line-of-sight and network path.

The minimum value of the last n RTT measurements is chosen as it reflects the propagation delay the best.

While seeming a good idea at first, the inclusion of all intermediates is not appropriate: First, rarely observed intermediates will not improve the estimation quality due to not having a sufficiently large neighborhood. Second, intermediates at the edge of the network are typically only observed on paths from and towards a specific node, thus not feeding the system with valuable positioning information. Third, the overall scalability in terms of computational effort will suffer from a large set of intermediate positions to be calculated. Hence, it is expedient to reduce the set of involved intermediates, e.g., to intermediates which are recognized on at least 1% of the observed paths' non-edge sections. As a practical result, the participating overlay nodes additionally calculate and update intermediate positions for the most relevant points of presence, e.g., the large Internet exchange points located at Frankfurt and Amsterdam.

VI. Evaluation

To assess the proposed approach, we will first discuss the fulfillment of the objectives from Sec. II. In a second part, the accuracy of localization is evaluated with the help of a simulation study.

No sole dependence on external location information: The proposed system components do not depend on the amount nor the quality of some specific external location hints. Nonetheless, to provide meaningful results, some nodes with access to geo-information are required. Given scenarios without any position hint, the system can only serve relative geographic distance information.

Scalability: Like in Vivaldi, the number of overlay neighbors for each node remains constant. The computational effort due to involvement of position hints can be considered negligible as nodes are practically aware of no more than one position hint. If, contrary to expectations, a specific node is aware of more than a few location hints, it may simply select a small random set. The involvement of intermediates causes additional measurement and computational effort, but the number of observable intermediate systems on end-to-end-paths is restricted to real-world transport network path lengths. Furthermore, the number of intermediates is rapidly reduced by only considering intermediates which are recognized on a significant amount of paths. Thus, the induced computational and measurement overhead can be considered being tolerable. Moreover, all mechanisms depend on local knowledge only. Hence, the system will scale over the number of participating nodes.

Robustness: The proposed system does not introduce exposed entities and tolerates fail-stop errors. Furthermore, the deployed fuzzy anchors do not only limit propagation of failures introduced by misleading position hints, but also offer the location of nodes with inaccurate or erroneous external information.

Absolute and relative Accuracy: The perhaps most significant objective regarding this article is evaluated with the help of quantitative measurements: To do so the occurring absolute geolocation error is evaluated for different fractions of fuzzy anchors. However, various applications, e.g., overlay backup path planning, do not only profit from absolute positions, but also from relative positioning. Hence, the relative position in reference to other peers is also compared to the real-world equivalents.

Simulative experiments were performed utilizing a previously measured *PlanetLab* dataset that consists of delays and traceroutes between a set of North American and European sites. This allows a determination of the occurring location errors as the reference position of each measurement site is publicly available. The proposed approach is simulated on top of this real-world dataset in a very detailed simulation environment, which is based on OMNeT++ [22] and the INET extension. Relevant parameters are the number of nodes (195), the sphere radius $r = 0.14s$ which reflects the mapping between distance and delay, and the number of neighbors n=32. All of the following charts show means over 100 simulation runs and the according 99% confidence intervals.

A. Accuracy of Geolocation Estimation

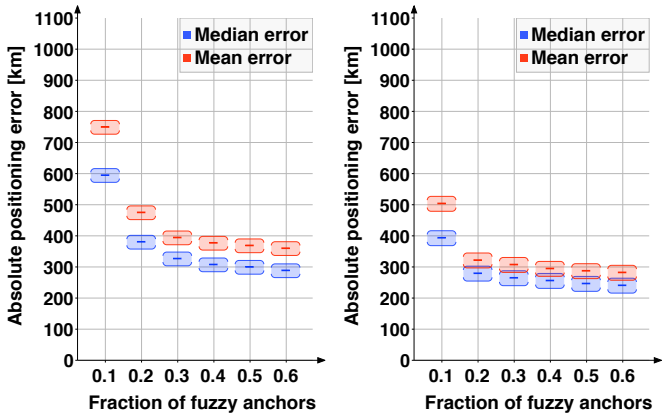
The most intuitive metric for location techniques is the resulting geographic position failure e_{geo}, which is the absolute distance between real-world positions i_{act} and estimated peer positions i_{est}. Additionally, to compare the relative positioning between all nodes, we perform an evaluation of the average distance error e_{dist}. To do so, the distance between predicted node positions $d(i_{est}, j_{est})$ and the distance between real positions $d(i_{act}, j_{act})$ is compared for pairs of nodes i, j:

$$e_{dist}(i,j) = |d(i_{act}, j_{act}) - d(i_{est}, j_{est})|$$

Given the set of all nodes $\mathcal{N}$, the average distance error e_{dist} of a node i is:

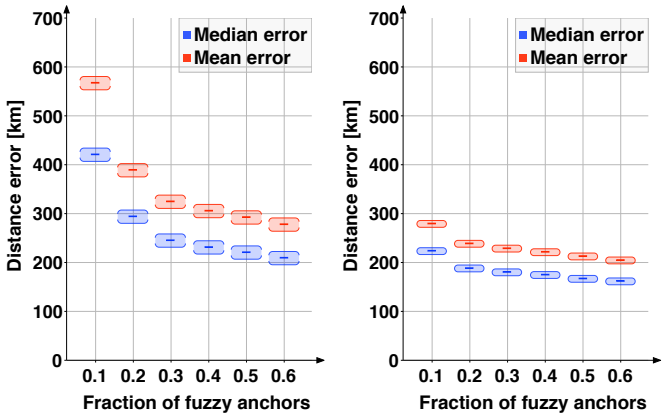
$$e_{dist}(i) = \frac{1}{\mathcal{N}-1} \sum_{j \in \mathcal{N}, i \neq j} e_{dist}(i,j)$$

Please note that, in order to only reflect the performance of the localization system itself, we regarded only nodes for $\mathcal{N}$ that do not have access to position hints.



(a) Global dataset (b) North American dataset

Fig. 6. Absolute geographic positioning error



(a) Global dataset (b) North American dataset

Fig. 7. Distance error

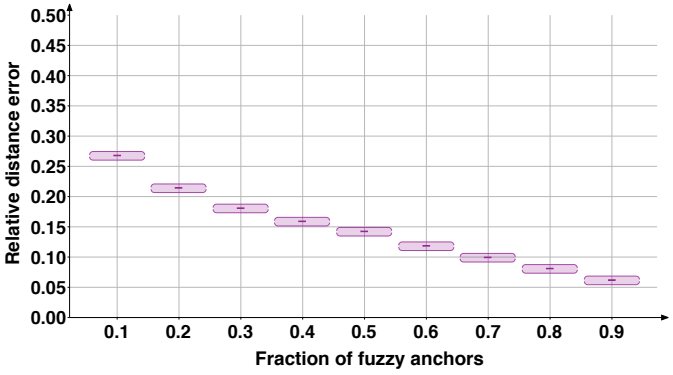


Fig. 8. Median of relative distance error for global dataset

Fig. 6(a) and 6(b) show the mean and median error of peer localization e_{geo}. While Fig. 6(a) shows the results for the dataset including North American and European nodes, the right-hand figure (6(b)) depicts results for the set of North American nodes only. Both scenarios confirm the intuition, that a minimum ratio of fuzzy anchors is required to obtain valuable localization. However, already a fair ratio of only 30% fuzzy anchors, the absolute estimation error decreases significantly in both scenarios. The gap between both scenarios can be widely attributed to a single issue: transatlantic connections cause significant smaller delays when compared to landlines of similar length as overhead through switching and layer 2 processing is negligible and even the according physical layer submarine cables are running comparably beeline-like. As a result, the distances of peer connections across continents are generally underestimated, disturbing the positioning accuracy of the overall system. However, this effect can be mitigated by periodically choosing new neighbors with a distance-based bias.

For comparison purpose, Fig. 7(a) and Fig. 7(b) illustrate the mean and median of the average distance error e_{dist} over all nodes that did not serve as a fuzzy anchor. The depicted shapes are similar to the absolute error, but the accuracy is comparably higher for both datasets. This indicates and leads to another issue that is to be mentioned: As the spring-mass algorithm substantially minimizes the embedding error of coordinates, there may be multiple optimal locations in terms of embedding error, leading to positioning errors in the sense of absolute localization. In particular, this is the case in sparsely populated regions and is similar to effects of traditional multilateration. Thus, the relative error can be expected to be lower than the absolute error, which is of importance to services like backup routing.

B. Relative Distance Error

While the achievable absolute and relative positioning accuracy cannot compete with the accuracy of GPS-based approaches, and is therefore not usable for applications that require an accurate absolute positioning, the accuracy may be acceptable for applications that rely only on relative comparisons between peers. In order to evaluate the error in reference to the connection distance, we additionally perform an evaluation of the relative distance error e_{rel}, which normalizes the distance error by the minimum of either the estimated or real distance. The minimum metric ensures that underestimation and overestimation are equally treated. Given the absolute distance error $e_{dist}(i,j)$ between two nodes i and j, e_{rel} is formally defined by:

$$e_{rel}(i,j) = \begin{cases} \dfrac{e_{dist}(i,j)}{d(i_{act},j_{act})} & \text{if } d(i_{act},j_{act}) \leq d(i_{est},j_{est}) \\ \dfrac{e_{dist}(i,j)}{d(i_{est},j_{est})} & \text{else.} \end{cases}$$

Fig. 8 shows the median distance error over all node pairs of the dataset including North American and European nodes. The depicted graph illustrates considerably small relative errors – even for small ratios of fuzzy anchors. Again, given ratio of around 30% fuzzy anchors, the error decreases considerably.

Taking everything into account, the results can be contemplated from two perspectives. On the one hand, the achievable accuracy is limited, so that it does not fit a range of typical location applications and cannot compete with external location services, e.g., GPS. On the other hand, given applications with less stringent requirements regarding position accuracy, e.g., verifying third-party database location services by detecting substantial outliers or serve matchmaking applications, the proposed method can be considered valuable. Even in

environments without access to location services, for GPS being not provided or geolocation database informations not available which is common, e.g., in sparsely populated areas, the presented approach can provide a rough estimate while solely relying on delay measurements. Furthermore, given a network of peers, estimations may be given for other network participants that do not cooperate, e.g., by not sending position information.

VII. Conclusion

Within this article, we studied the question, how well a fully distributed approach can estimate the geographic positions of peers by delay measurements. Given a real-world dataset with peers on multiple continents and a fair amount of external position hints of about 30%, we illustrated that absolute peer positions can be estimated with an accuracy of less than 350 kilometers in the average case. With regard to denser populated overlay networks, the accuracy can be expected to advance. While the achievable accuracy is not comparable to external geolocation services, e.g., GPS, it still satisfies the requirements of services like the planning of backup routes by solely depending on delay measurements. Furthermore, it is conceptually complementary to the widespread IP mapping databases, thus giving opportunities for both verifying the error-prone database lookups and providing location estimations for sparsely populated areas, which are seriously underrepresented in IP location databases.

There are plans to increase the localization accuracy by additional extensions in future research, e.g., periodic cluster-based neighborhood selection. In addition, the spring-mass embedding will be compared to alternative embedding techniques based on equation solving and least squares optimization. Furthermore, we want to evaluate the effectiveness of the proposed approach by interacting with overlay applications for robust backup path selection based on geographic node positions.

References

[1] S. Agarwal and J. Lorch. Matchmaking for online games and other latency-sensitive P2P systems. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 315–326, 2009.

[2] M. Arif, S. Karunasekera, S. Kulkarni, A. Gunatilaka, and B. Ristic. Internet host geolocation using maximum likelihood estimation technique. In *Conference on Advanced Information Networking and Applications (AINA)*, pages 422–429, 2010.

[3] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem. Analysis of End-to-end Delay Measurements in Internet. In *Proc. of the Passive and Active Measurement Workshop-PAM'2002*, 2002.

[4] D. Choffnes and F. Bustamante. Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-peer (P2P) Systems. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 363–374. ACM, 2008.

[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *ACM SIGCOMM*, pages 15–26, 2004.

[6] C. Davis, I. Dickinson, T. Goodwin, and P. Vixie. A means for expressing location information in the domain name system. 1996.

[7] Z. Dong, R. Perera, R. Chandramouli, and K. Subbalakshmi. Network measurement based modeling and optimization for IP geolocation. *Computer Networks*, 56(1):85–98, 2012.

[8] F. Girlich, M. Rossberg, G. Schaefer, T. Boehme, and J. Schreyer. Bounds for the Security of the Vivaldi Network Coordinate System. *Accepted for Conference on Networked Systems (NetSys)*, 2013.

[9] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-Based Geolocation of Internet Hosts. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 288–293. ACM, 2004.

[10] M. Gunes and K. Sarac. Analytical IP alias resolution. In *IEEE ICC*, volume 1, pages 459–464. IEEE, 2006.

[11] E. Katz-Bassett, J. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84. ACM, 2006.

[12] J. Ledlie, P. Gardner, and M. Seltzer. Network Coordinates in the Wild. In *4th USENIX Symposium on Networked Systems Design & Implementation*, pages 299–311, 2007.

[13] J. Liu, B. Priyantha, T. Hart, H. Ramos, A. Loureiro, and Q. Wang. Energy Efficient GPS Sensing with Cloud Offloading. 2012.

[14] J. Muir and P. Oorschot. Internet geolocation: Evasion and counterevasion. *ACM Computing Surveys (CSUR)*, 42(1):4, 2009.

[15] S. Neumayer, A. Efrat, and E. Modiano. Geographic Max-Flow and Min-Cut Under a. Circular Disk Failure Model. In *INFOCOM, 2012 Proceedings IEEE*, pages 2736–2740. IEEE, 2012.

[16] V. Padmanabhan and L. Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 173–185. ACM, 2001.

[17] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye. IP geolocation databases: Unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

[18] M. Rossberg, R. Golembewski, and G. Schaefer. Attack-Resistant Distributed Time Synchronization for Virtual Private Networks. In *IEEE ICCCN*, 2012.

[19] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, et al. Detour: Informed Internet Routing and Transport. *IEEE Micro*, 19(1):50–59, 1999.

[20] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. *IEEE/ACM Transactions on Networking*, 12(6), 2004.

[21] Y. Shavitt and N. Zilberman. A Geolocation Databases Study. *IEEE JSAC*, 29(10):2044–2056, December 2011.

[22] A. Varga et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, volume 9. sn, 2001.

[23] T. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.