Shortcuts in a Virtual World

Moritz Steiner[*] and Ernst W. Biersack
Institut Eurécom
06904 Sophia–Antipolis
France
{steiner,erbi}@eurecom.fr

ABSTRACT

We consider the case of a virtual world of peers that are organized in an overlay built by Delaunay Triangulation. Application layer routing is used to determine the path taken in the overlay between two peers. Application layer routing incurs a major delay penalty since it ignores the characteristics of the physical network topology.

We show how to augment a Delaunay based overlay by a small and bounded number of additional links called *shortcuts*. A peer chooses its shortcuts among the nodes that are physically close to him in the underlay while covering at the same time uniformly the overlay space. Shortcuts improve the average hopcount and the average delay for a path between two peers from $\mathcal{O}(N^{1/d})$ to $\mathcal{O}(\log(N))$, where N is the total number of peers in the overlay and d the dimension of the overlay. The algorithm to manage shortcuts is fully distributed and requires only local knowledge.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network topology, Distributed networks*

General Terms

Algorithms, Performance

Keywords

P2P, overlay, small world, routing

1. INTRODUCTION

Networked Virtual Environments (NVEs) are computer generated synthetic worlds that allow simultaneous interactions between multiple participants. Especially with the boom of Massive Multiplayer Online Games (MMOGs), NVEs are becoming increasingly popular. To create a large scale NVE, the traditional client-server model does not scale and a peer-to-peer based approach is required that constructs an overlay connecting all the participants.

[*]Moritz Steiner is also affiliated with the Department of Mathematics and Computer Science, University of Mannheim, Germany.

NVEs have several requirements that influence the choice of the overlay

- Peers must be able to freely choose their peerId, which in fact reflects their position in the overlay
- Peers move around and their position and peerId will change
- The overlay must efficiently support the communication of a peer with its close-by neighbors.

There exist a large number of structured overlays such as Pastry [16], Tapestry [23], Chord [19], CAN [15]. However, these overlays typically assign to each peer a fixed peerId, for instance the hash of the peer's IP address, which is not appropriate for NVEs.

Delaunay triangulation [5], on the other hand, meets all these requirements. Structured overlays based on Delaunay triangulation have been studied previously [11, 18]. A Delaunay based overlay organizes peers according to their *position* in the NVE. However, as do most overlays – a Delaunay based overlay completely ignores the position of the peers in the physical network. As consequence, two peers that are neighbors in the overlay may be physically far away and any message exchange along a sequence of overlay hops will experience a significant *delay penalty*.

To reduce the delay penalty of overlay routing, we propose to augment each node of the Delaunay based overlay with a limited number of carefully selected shortcut links.

The rest of this paper is organized as follows: Section 2 introduces definitions necessary for the description of our algorithm and section 3 discusses the related work. Section 4 presents the algorithms for shortcut selection and shortcut routing, called *Shortcut Augmented Overlay* (SAO). In Section 5 we evaluate SAO and present some simulation results. Finally, Section 6 concludes the paper.

2. DEFINITIONS

2.1 Terminology

An overlay network is a network that is built on top of another network, called the **underlay** or the **physical network**. Let $\mathcal{N}$ denote the set of all nodes in the underlay and $\mathcal{O}$, $\mathcal{O} \subset \mathcal{N}$ the set of nodes in the overlay. The underlay consists of a set of nodes $\mathcal{N}$ (routers and end hosts) connected by *physical* links. The overlay consists of a set of nodes $\mathcal{O}$ (end hosts only) connected by *virtual* links.

A peer p, $p \in \mathcal{O}$, has two types of coordinates: (i) **overlay** coordinates p^o that indicate the position of p in the NVE and (ii) **network** coordinates p^u that reflect the position of the peer in the underlay. The network coordinates can be calculated at very low cost using a network coordinate system such as Vivaldi [4]. Network coordinates are used to compute the underlay distance between two peers.

Given two peers p and k, $d^o(p,k)$ denotes the **overlay distance** between p and k, which is defined as the Euclidean distance between the virtual coordinates p^o and k^o. Similarly, $d^u(p,k)$ denotes the **underlay distance** between p and k, which is defined as the Euclidean distance between the network coordinates p^u and k^u. Overlay distance is a measure of the *routing overhead* since at each overlay hop some processing must be done, whereas underlay distance is a measure of the *delay* a message will experience before reaching the destination. If underlay distance between two nodes p and k is less than a threshold d_t, i.e. $d^u(p,k) < d_t$, p and k are considered to be very close to each other in the underlay and are called **physical neighbors**.

We have now all the necessary notation to explain **delay penalty** more concisely: Consider two peers p and k. Let $p = h_0, \ldots, h_i = k$ be the overlay peers visited by a message sent from peer p to peer k. The path of *overlay* hops visited may be much longer than the distance separating the two peers in the underlay, a case which we refer to as delay penalty:

$$\sum_{j=0}^{i-1} d^u(h_j, h_{j+1}) >> d^u(p,k).$$

2.2 Delaunay triangulation

Since the goal of a NVE is to create a virtual world that is as realistic as possible, a 3-dimensional representation of the NVE is required.

Given n 3-dimensional points that represent to positions of our peers in the NVE, a 3-dimensional Delaunay triangulation (DT) connects the points into non-overlapping tetrahedra that fill the convex hull of these points. If all point pairs whose region share a common plane are joined by straight lines, the result is a triangulation of the convex hull of the n points. This triangulation is known as Delaunay triangulation[1]. See figure 1 for a Delaunay triangulation in 2 dimensions.

[1]The dual of the Delaunay triangulation is the Voronoi Diagram [22], which assigns to each of the n points a region that is nearer to that point than to any other point.

The points represent the peers in the NVE and the edges the links between peers. The nodes in the overlay that are connected with peer p via a virtual link are referred to as **overlay neighbors** of peer p, denoted as $V(p)$, $V(p) \subset \mathcal{O}$.

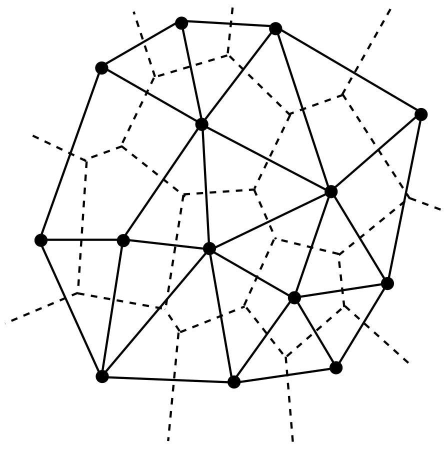


Figure 1: Delaunay Triangulation (solid lines) and Voronoi diagram (dashed lines) in 2 d.

Overlay routing is done in a greedy fashion: The message is forwarded to the overlay neighbor that reduces the remaining overlay distance to the destination most among all neighbors. In Delaunay based overlays, greedy routing requires $\mathcal{O}(N_o)$ time in the worst case and $\mathcal{O}(N_o^{1/d})$ in the average case, where N_o denotes the number of the nodes in the overlay, and d the dimension of the overlay.

To improve the performance of overlay routing, additional overlay links, called shortcut links are introduced. The overlay nodes to which a peer p has a a shortcut link with peer p are called **shortcuts** of peer p, and denoted as $S(p)$, $S(p) \subset \mathcal{O}$.

3. RELATED WORK

When structured overlays where first introduced, very little attention was payed to exploit information about the underlay-proximity of nodes for overlay routing. As a consequence, a message sent to a destination using overlay routing will incur a high delay penalty. To reduce the delay penalty, various proposal were made. In CAN [15], each node measures its underlay distance to a set of landmark nodes, in an effort to determine its relative position in the Internet and to construct an Internet topology aware overlay. Tapestry and Pastry use prefix-based routing, which leaves considerable freedom in the choice of the overlay neighbors.

Pastry can be enhanced by a proximity neighbor selection [3] where a node p selects among all the possible nodes with the appropriate prefix the node with the smallest underlay distance from node p. The shorter the prefix to be matched, the more candidate nodes with the appropriate prefix exist and the smaller the underlay distance between the closest node and p. If proximity neighbor selection is applied, the overlay path $p_i, p_{i+1}, \cdots, p_{i+h-1}, p_{i+h}$ taken by a the message will be such that for any two consecutive overlay hops, the overlay distance will decrease, i.e. $d^o(p_l, p_{l+1}) > d^o(p_{l+1}, p_{l+2})$, while the underlay distance will increase, i.e. $d^u(p_l, p_{l+1}) < d^u(p_{l+1}, p_{l+2})$. In practice the underlay distance $d^u(p_{i+h-1}, p_{i+h})$ of the last hop will dominate the total distance traveled by that message .

There exists another class of overlays that try to exploit the characteristics of small world graphs to improve search efficiency. The notion of small world phenomenon originates from social science research [14, 20, 21] and has found a lot of interest in the physics, computer science, and mathematics community. Observations indicate that the small world phenomenon is pervasive in a wide range of settings such as social communities, biological environments, and communication networks.

Informally, a small world network can be viewed as a connected graph in which two randomly chosen nodes are connected by a short path through the graph with large probability. The term *small world effect* means that the average distance between two nodes in the network increases logarithmically with the total number of nodes in the network.

Kleinberg [8] presented an algorithm to build a small world graph that augments a graph consisting of a grid by a constant number of additional long range links. However, to create the additional links, global knowledge is required. For message forwarding, a greedy routing procedure is used where each node forwards the message to the neighbor that is the closest to the destination.

Approaches based on the small world phenomenon [7, 10, 13, 12] have a number of limitations. They do not take into account the physical network topology and they do not allow the peers in the NVE to freely choose and change their position.

In summary, we have seen two different approaches to improve overlay routing: One approach is to better match the overlay and the underlay in order to reduce the delay penalty and another one is to construct a small-world overlay to keep the number of overlay hops traveled small. However, up to now these two approaches have not been explored together as will be done in in this paper with SAO.

4. ALGORITHMS FOR A SHORTCUT AUGMENTED OVERLAY

SAO aims to improve overlay routing of messages by introducing shortcut links. More precisely

- SAO reduces the number of hops in the overlay by augmenting the overlay with additional links in such a way that the overlay resembles a small world: Each peer p has a limited number of shortcuts $S(p)$ that "cover the whole virtual world". These shortcuts $S(p)$ are organized into concentric, non-overlapping rings that are divided into 2^d quadrants (Figure 2) with peer p is at the center. For a given number of rings R, the maximum number of shortcuts is $2^d * R$, where d is the dimension of the overlay.

- SAO reduces drastically the delay penalty since shortcuts are chosen among the *physical neighbors.*

4.1 Shortcut Insertion

Our aim is to make shortcut insertion as lightweight as possible. For this purpose SAO uses a lazy algorithm to insert new shortcuts: A peer continuously learns about the existence of other peers, (i) during the join procedure, (ii) while traveling the virtual world, (iii) or simply when forwarding a message.

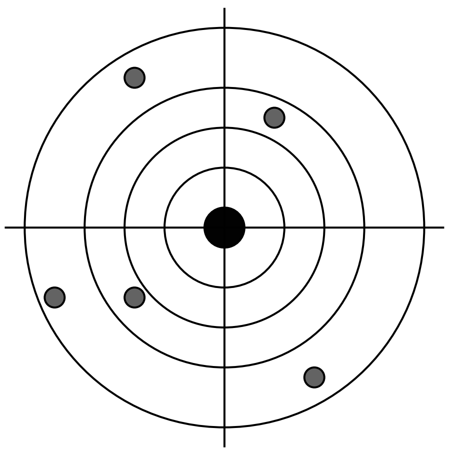


Figure 2: Structure (for 2 dimensions) used to organize the shortcuts. The dark point in the center represents peer p, the grey points are the shortcuts $S(p)$ of peer p. Note that not all the possible fields are used.

Whenever a node k qualifies as shortcut (algorithm 1: line 1) the corresponding **field** — defined by the number of the ring r and the number of the quadrant q — is computed, and k is added to the list of shortcuts (lines 2-3). Any shortcut previously stored at that field will be overwritten. In doing so, we will constantly "renew" the shortcuts. This reduces the probability of stale shortcuts without having to check periodically if they are alive.

Whenever k is a shortcut for p, p is also a shortcut for k. Therefore, p informs k to add p to its shortcuts (line 4).

Algorithm 1: addShortcut(k) (executed by peer p to add peer k to $S(p)$.)

```
1 if $d^u(p,k) < d_t$ then
2     $r, q \leftarrow$ getRelativePosition($k$)
3     $S(p) \leftarrow k, [r, q]$
4     $k$.addShortcut($p$)
```

When a node p joins the overlay first, its set of shortcuts will be empty. In this case p may simply ask some of the nodes it has encountered during the join for their shortcuts. This means that the peers are *learning* from each other and find shortcuts while communicating.

We have seen that local knowledge only is sufficient to find the shortcuts. This and the fact that shortcuts are physical neighbors is the major difference between SAO and the algorithm of Kleinberg.

4.2 Shortcut Routing

In SAO, a node p has two types of neighbors: overlay neighbors $V(p)$ and shortcuts $S(p)$. The nodes in $V(p)$ are close to p in the *overlay*, whereas the nodes in $S(p)$ are close to p in the *underlay.*

The idea is to use shortcuts whenever possible and to rely on overlay neighbors only when no shortcuts are available.

Consider a message to be sent to destination d. When the routing procedure computes the next hop, it first tries to find a shortcut $s \in S(p)$ that minimizes the remaining distance in the overlay $d^o(s,d)$ while incurring as little delay in the underlay as possible. For this purpose, the ring r and the quadrant q of d are calculated (Algorithm 2, line 1). In case the field is empty, the fields in the same quadrant of interior rings (i.e. rings closer to the center) are checked (line 3-5). While shortcuts stored in interior rings $r-1,\ldots,1$ will not reduce the remaining overlay distance as much as a shortcut in ring r, they will reduce the remaining distance more than any overlay neighbor $V(p)$. If no appropriate shortcut is found, the overlay neighbor in $V(p)$ closest to the destination d is selected as next hop (line 6-10).

Using this kind of routing, a message will first use shortcut and travel long overlay distances but short underlay distances and as it approaches the destination it will use overlay neighbors and travel only small overlay distances but large underlay distances. Remember, the same behavior was achieved in Pastry when using proximity neighbor selection (see Section 3).

Algorithm 2: findShortcut(d) (executed by peer p searching a route to destination d.)

```
1  $r, q \leftarrow$ getRelativePosition($d$)
2  $closest \leftarrow null$
3  while $closest = null$ and $r > 1$ do
4      $closest \leftarrow S(p)[r,q]$
5      $r \leftarrow r - 1$
6  if $closest = null$ then
7      $closest \leftarrow p$
8      foreach $n \in V(p)$ do
9          if $d^o(n,d) < d^o(closest,d)$ then
10             $closest \leftarrow n$
11 return $closest$
```

5. SIMULATIONS

We carry out simulations to evaluate the performance improvement due to shortcuts. We show that a small number of shortcuts is sufficient to significantly decrease the number of hops and the delay of the path taken: The expected number of overlay hops and the expected delay[2] of a path are no longer $\mathcal{O}(N_o^{1/d})$ but can be reduced to $\mathcal{O}(\log(N_o))$.

5.1 Simulation Setup

The overlay and the underlay needs to be simulated together. We will first explain how to create the underlay and the overlay and then how to assign overlay nodes to the underlay.

Underlay: GT-ITM

Gt-itm [2, 1] is a widely-used tool to create synthetic network topologies. We used it to generate a 2–tier topology that consists of interconnected domains, and nodes inside each domain (Figure 3). Gt-itm provides us with two dimensional coordinates for each node and the links between them. The coordinates are used as network coordinates indicating the position of a node (in the underlay).

End users that may participate in the overlay usually have one link only. Therefore a big fraction of nodes must have only one link.

[2]In the following we will use the terms distance and delay interchangeably.

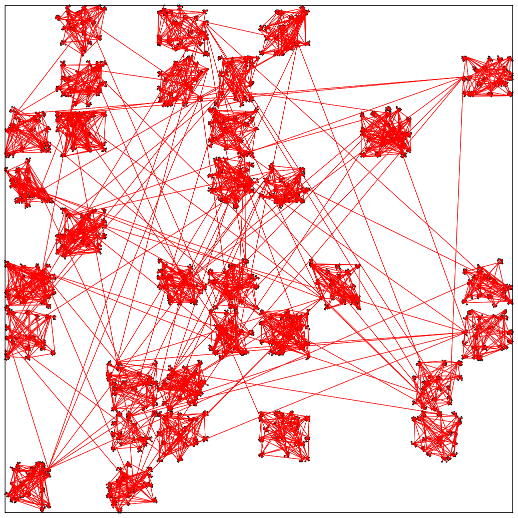


Figure 3: Network topology with 8000 nodes organized in 2–tiers generated with gt–itm.

For our simulations we need to choose the threshold d_t that is used to determine whether two nodes are physical neighbors. Figure 4 shows the histogram of the underlay distances. In our case values around 80-100 ms are reasonable for d_t, thus that only nodes lying in the same physical domain can be chosen as shortcuts. Note that the threshold has not to be adapted to the number of nodes in the underlay N.

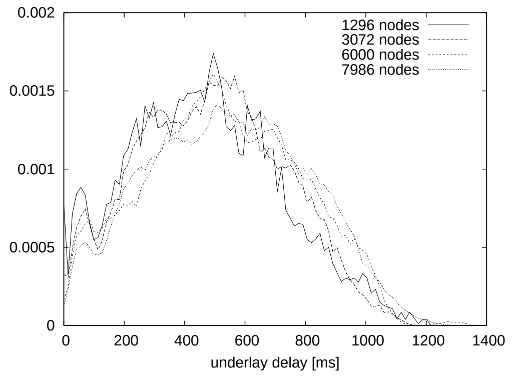


Figure 4: Underlay delay distribution between all pairs of nodes for topologies generated with different number of nodes N.

Overlay: DTON

The overlay used for the simulation uses DTON [17], a fully distributed P2P overlay network based on 3-dimensional Delaunay triangulation.

The node distribution in a NVE is not uniform since nodes are organized in clusters. To obtain these clusters, we use the so called Lévy Flight [9] that produces a random walk through space using the Lévy distribution [6] to determine the step size. The angles at the turning points are randomly chosen. Removing the path and looking only at the (turning) points, we get a distribution of points organized in smaller and bigger clusters (Figure 5).

We ran simulations with uniform and clustered node distributions.

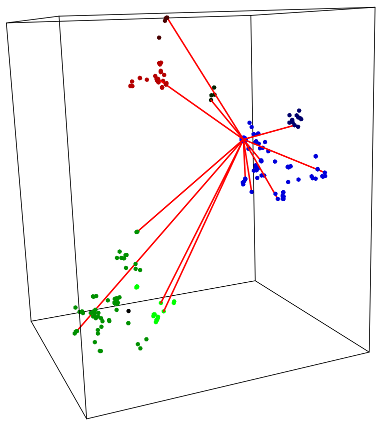


Figure 5: Node distribution obtained with Lévy Flight.

Assignment of overlay nodes to underlay

Participants in NVEs are humans situated at the edge of the Internet. Therefore, among the nodes generated only those with a single link may participate in the overlay. We simulated scenarios where different fractions of those nodes that participate in the overlay, but did not observe that the fraction of participants had a measurable influence on the results obtained.

The assignment between the nodes of the generated network and the overlay nodes is chosen random uniformly, which will assure that peers located in a same network domain are well distributed in the overlay.

5.2 Metrics

The two most important metrics to assess the improvement due to shortcuts are the number of hops and delay experienced by a message.

Delay is best suited to compare the performance of overlay routing, since the users strongly care about the perceived delay.

The memory overhead needs to be measured too, which will be done counting the average number *shortcuts per node*:

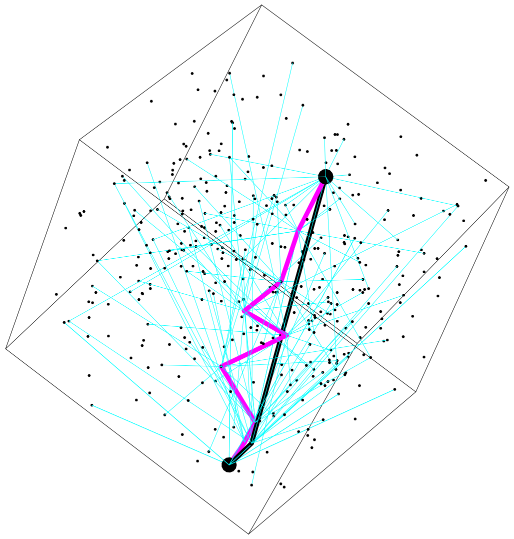
$$S_{N_o} = \frac{\sum_{i=1}^{N_o} |S(i)|}{N_o}.$$

The value of S_{N_o} depends on d_t, since only if $d^u(p,k) < d_t$ peers p and k may consider each other as shortcut. Increasing d_t leads to an increasing number of shortcuts per node. S_{N_o} also depends on the number of rings R (see Figure 2). If each field is occupied, $2^d R$ shortcuts are stored per node. Therefore, $S_{N_o} \leq 2^d R$.

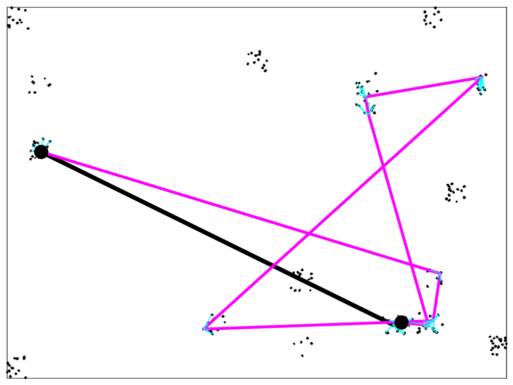
Since the peers get to know shortcuts by being part of the overlay, N_o messages are sent between random peers before the performance measures are computed. This ensures that most peers know some shortcuts.

5.3 Results

Let us first illustrate how SAO works. The example displayed in figure 6 contains 1296 nodes and 3680 edges in the underlay and 397 nodes participating in the overlay. The path displayed is the longest one, which benefits most of the shortcuts.



(a) Path taken in the overlay, from the point at the top to the point at the bottom. The thin lines indicate the shortcuts of the nodes visited.



(b) Path taken in the underlay, from the point on the right to the point on the left. Grey lines: path without shortcuts; dark lines: path with shortcuts.

Figure 6: Path taken in the overlay and in the underlay.

Considering the *overlay* (Figure 6(a)), what is the difference between the paths that use the shortcuts and the ones that do not? With shortcuts, there exist overlay hops to more distant nodes and therefore the number of hops per path is smaller. However, this does not say anything about the routes taken in the underlay and their respective delay.

Considering the *underlay* (Figure 6(b)), priority is given to choosing overlay hops that are nearby in the underlay. If we compare the two different paths at the underlay level, we see that shortcuts can be very effective: The path without shortcuts uses 6 overlay hops and has a total delay of 11049 ms. Using the shortcuts, these values are reduced to 3 hops and 3418 ms. For comparison, shortest path routing in the underlay gives 2471 ms. Only the first of the three hops used is a shortcut (it is the very long one in figure 6(a)).

Figure 7 shows the shortcuts of a node. We see that shortcuts are distributed over the entire region.

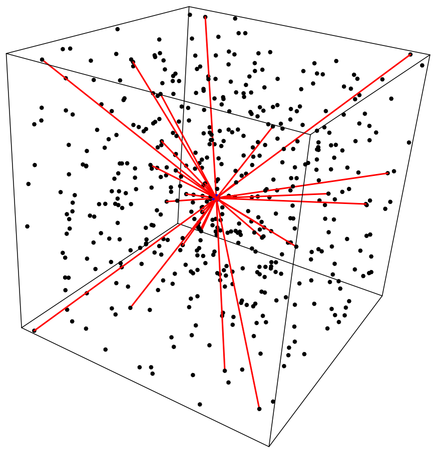
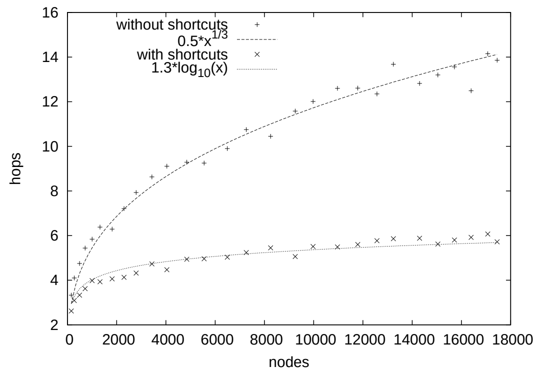


Figure 7: The lines show the shortcuts of the node having most shortcuts.

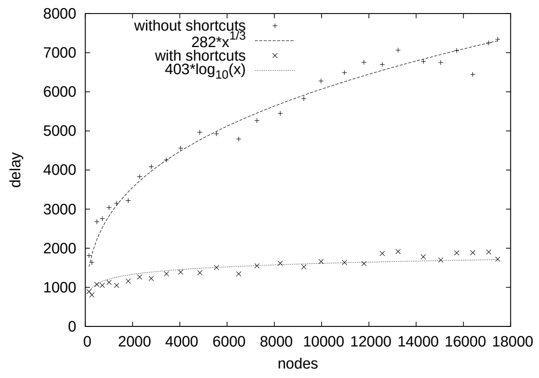
One goal of using shortcuts is to reduce the number of hops in the overlay. As noted earlier, the expected number of overlay hops in an overlay based on the DT is $\mathcal{O}(N_o^{1/d})$, for the three dimensional implementation used it is $\mathcal{O}(N_o^{1/3})$. Using shortcuts, the expected number of hops can be reduced to $\mathcal{O}(\log(N_o))$ (Figure 8(a)). The same is true for the delay (Figure 8(b)).

To control the memory requirement introduced by the shortcuts, the threshold d_t can be adjusted. Increasing d_t leads to more shortcuts per node but also to smaller hop counts and delays. Adding only a few shortcuts already has a great impact (Figure 9). After a certain point, adding more shortcuts does not reduce the hop count or the delay significantly anymore. The sweet spot lies around $d_t = 80$. The sweetspot directly depends on the network topology. In the topology generated with gt-itm, the domains have a diameter of about 100ms. Therefore it does not bring much improvement to choose a threshold bigger than $d_t = 100$.

Among all possible paths the very short ones with only one or two overlay hops, which will not benefit a lot (or even not at all) from the shortcuts. However, particularly the long paths with at least 10 hops benefit a lot from shortcuts. In figure 10 the complementary cumulative distribution function (CCDF) of the number of hops and delay is plotted. We see that shortcuts significantly reduce both, the number of hops and the delay.



(a) Average overlay hop count.



(b) Average delay.

Figure 8: Average hop count and delay of 100 randomly chosen paths, depending on the size N_o of the overlay network. ($d_t = 80$ ms)

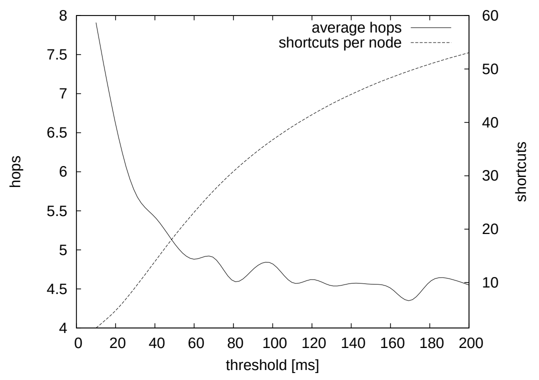
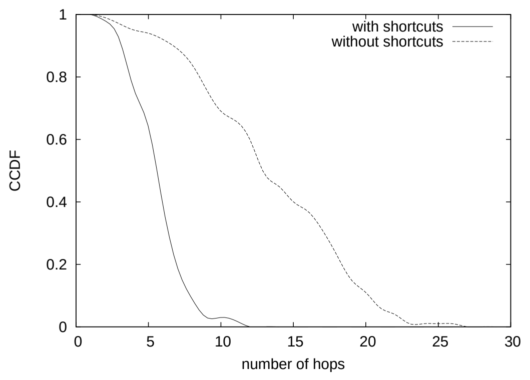


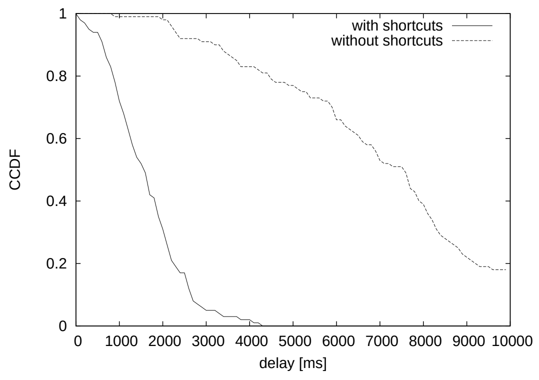
Figure 9: Number of shortcuts per node and the average hop count as function of threshold d_t ($N_o =$ 18000 nodes).

In figure 10(a), for instance, we see that without shortcuts 20% of the longest paths have between 18 and 27 hops, while with shortcuts 20% of the longest paths have only between 7 and 12 hops.

While not all nodes have the same number of shortcuts, the variation in the number of shortcuts is small as can be seen in figure 11(a), which depicts the CCDF of the fraction of shortcut fields that are filled with shortcuts for different

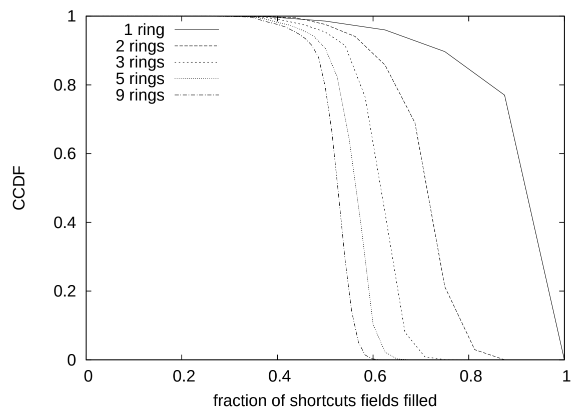


(a) The distribution of the overlay hop count.

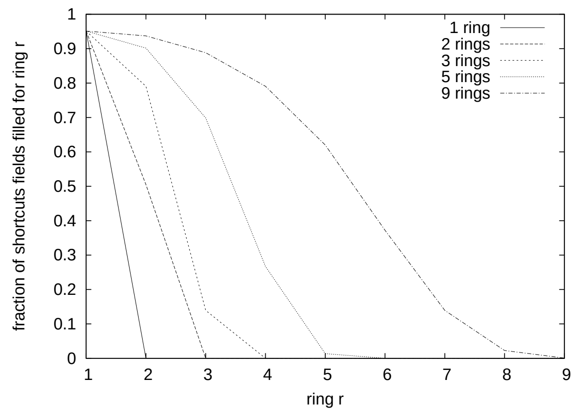


(b) The distribution of the delay.

Figure 10: Complementary cumulative distribution function of hop count and delay for 100 randomly chosen paths ($N_o = 18000$ nodes, $d_t = 80$ ms).



(a) CCDF of the number of shortcuts per node.



(b) The distribution of the number of shortcuts per ring r, averages over all overlay nodes.

Figure 11: Distribution of the number of shortcuts per node and per ring for different numbers of rings R ($N_o = 18000$ nodes, $d_t = 80$ ms).

numbers of rings R. When we increase R, fewer fields are filled. For $R = 1$ nearly all nodes find a shortcut for each field.

The width of each ring depends on the number of rings, since the total space covered remains always the same. When doubling the number of rings R, the width of each ring is halved. Therefore, with increasing R, it will become less likely to find an appropriate shortcut for each field. This can be seen in both figures: With increasing R, the fraction of the fields filled decreases (figure 11(a)). With increasing ring number r, the fraction of the fields filled decreases (figure 11(b)), which means that rings towards the center have more fields filled with shortcuts than outer rings.

Figure 12(a) shows the evolution of the number of shortcuts per node depending on the number of rings. The average number of shortcuts grows logarithmically with the number of nodes. The absolute number of shortcuts remains in the order of a few tens, which means that the storage overhead will be very low.

With increasing R, the number of hops on a path decreases, since each node stores more shortcuts. Figure 12(b) shows the average number of hops for 100 randomly chosen paths. The first ring introduces the biggest benefit (compared to an overlay without shortcuts); with each additional ring the added benefit diminishes.

By adding more rings, each peer will have more shortcuts and therefore the average hop count and delay will be reduced. The evaluate the tradeoff between hop count reduction vs. additional storage cost, we plot in Figure 13 how, for each additional ring, the ratio between hop count reduction and the number of additional shortcuts introduced. We see that the ratio decreases rapidly for increasing R. We propose to set R to 3 or 4, which limits the shortcuts per node to 24 or 32 in the 3 dimensional case.

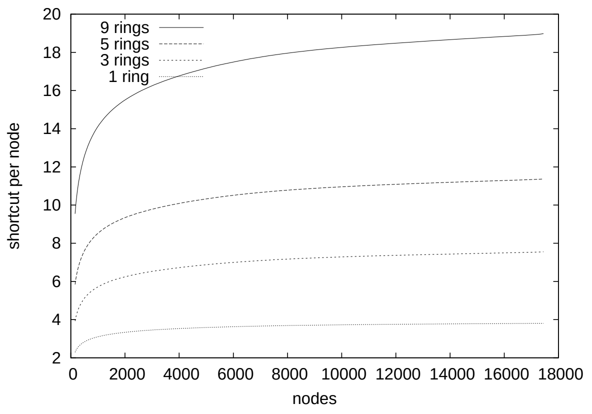
6. CONCLUSION

We showed how a Delaunay based overlay can be augmented in very simple way by additional shortcuts to build a small world overlay that will reduce both, average number of hops and delay. When choosing shortcuts, we exploit the fact that nodes are close to each other in the underlay may be far away in the overlay.

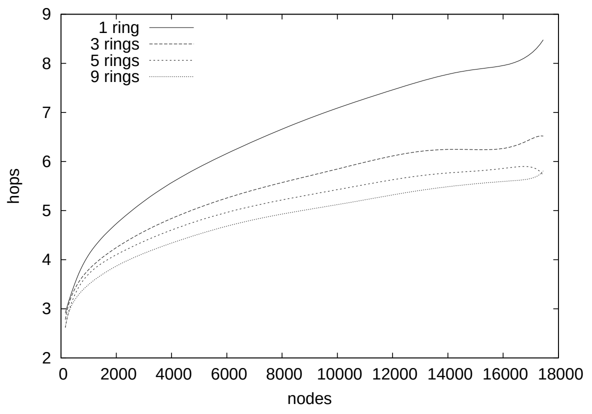
The idea of shortcuts is not limited to Delaunay based overlays but can be used in other structured overlays. A promising candidate will be CAN, where each node only knows overlay nodes that are close-by and shortcuts can be used to increase the overlay distance traveled at each hop.

Acknowledgement

We would like to thank one of the reviewers for the very detailed comments. This research was supported in part by France Telecom contract number CRE 46128283.



(a) The average number of shortcuts per node.



(b) The average overlay hop count.

Figure 12: Average number of shortcuts per node and average hop count of 100 randomly chosen paths, as function of the number of rings R ($N_o = 18000$ nodes).

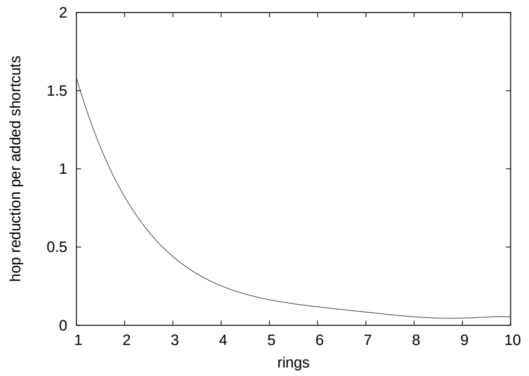


Figure 13: Reduction of the average overlay hops per added shortcut per node depending on the number of rings ($N_o = 18000$ nodes).

7. REFERENCES

[1] K. Calvert, M. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[2] K. Calvert, J. Eagan, S. Merugu, A. Namjoshi, J. Stasko, and E. Zegura. Extending and enhancing gt-itm. In *MoMeTools '03: Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 23–27, 2003.

[3] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, One Microsoft Way, Redmond, WA 98052, 2002.

[4] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings ACM SIGCOMM 2004*, Aug.

[5] B. Delaunay. Sur la sphère vide. A la mémoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7:793–800, 1934.

[6] M. Gutowski. Lévy flights as an underlying mechanism for global optimization algorithms. *ArXiv Mathematical Physics e-prints*, June 2001.

[7] K. Hui, J. Lui, and D. Yau. Small world overlay p2p networks. In *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on*, pages 201–210, 2004.

[8] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170, 2000.

[9] P. Lévy. *Théorie de l'Addition des Variables Aléatoires*. Gauthier-Villiers, Paris, 1937.

[10] M. Li, W.-C. Lee, and A. Sivasubramaniam. Semantic small world: An overlay network for peer-to-peer search. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)*, pages 228–238, 2004.

[11] J. Liebeherr, M. Nahas, and S. Weisheng. Application-layer multicasting with delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, Oct. 2003.

[12] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni. A distributed approach to solving overlay mismatching problem. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 132–139. IEEE Computer Society, 2004.

[13] S. Merugu, S. Srinivasan, and E. Zegura. Adding structure to unstructured peer-to-peer networks: The role of overlay topology. In *Proceedings of NGC 2003*, volume 2816, pages 83–94, Sept. 2003.

[14] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, may 1967.

[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM*, 2001.

[16] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale Peer-to-peer systems. In R. Guerraoui, editor, *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, LNCS, pages 329–350, Heidelberg, Germany, November 2001. Springer.

[17] M. Steiner. Structure and algorithms for the collaboration between peers and their application in

solipsis. Master's thesis, University of Mannheim and Institut Eurécom, Sophia-Antipolis, France, Mar. 2005.

[18] M. Steiner and E. W. Biersack. A fully distributed peer to peer structure based on 3d delaunay triangulation. In *Proceedings of Algotel - Septièmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, pages 93–96, May 2005.

[19] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, 2001.

[20] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–410, 1998.

[21] D. J. Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press, Princeton, NJ, USA, 1999.

[22] M. Yvinec and J.-D. Boissonnat. *Algorithmic Geometry*. Cambridge University Press, 1998.

[23] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, Apr 2001.