 The definitive version of this paper has been published in Workshop on Modeling, Simulation and Optimization of Peer-to-peer environments (MSOP2P) in conjunction with Euromicro (PDP 2007), 2007, 10.1109\/pdp.2007.42.

Efficient Simulation of Large-Scale P2P Networks: Modeling Network Transmission Times

Gerald Kunzmann*, Robert Nagel
Technical University of Munich
Institute of Communication Networks
Germany

Tobias Hoßfeld, Andreas Binzenhöfer
University of Würzburg
Institute of Computer Science
Germany

Kolja Eger
Hamburg University of Technology (TUHH)
Department of Communication Networks
Germany

Abstract

The ongoing process of globalization leads to a huge demand for highly scalable applications that are able to deal with millions of participants distributed all over the world. Peer-to-Peer (p2p) technology enables an arbitrary large number of users to participate in distributed services like content distribution or collaboration tools. In order to verify a new protocol's performance and scalability simulation is a commonly used tool. First, predicting the network and peer behavior in the real world is only feasible if the simulation, i.e. all applied models as well as the peer state, is as realistic as possible. Second, many properties of the system only become observable when the number of participants is sufficiently large. Therefore, verifying the scalability of a system requires simulating huge worldwide networks. Due to limited processing power, central memory and availabe time, both requirements can only be fullfilled if the applied models are very efficient. In this paper we take a closer look at the network layer. We compare the most commonly-used network models and present a very efficient model for applying real-world network transmission times in large scale simulations.

1 Introduction

Simulation, emulation and analytical approaches are the main methods in the process of developing and evaluating new networking protocols and applications. Especially in Peer-to-Peer (p2p) applications, where an arbitrary large number of users may participate in the network, simulation is required to prove the protocol's scalability and verify its correct behavior. Simulations are also used to evaluate the behavior of an application in uncommon or even undesirable situations, like the functionality of an ad-hoc network used for coordinating an action force after a natural disaster or simultaneuos failure of many peers participating in a p2p network. In order to achieve realistic results the models applied to the simulation must reflect the real world as close as possbile. However, the greater the level of detail, the more complex and resource consuming the model gets. Therefore, a good model is only as detailed as necessary.

Modeling the network is mandatory for networking applications. In many simulations, it is sufficient to use a model that adds a constant delay to all packets. In some cases transmission delay may even be neglected at all. In our current research on applying p2p mechanisms to Voice-Over-IP (VoIP) solutions we try to achieve a certain QoS by, e.g., accelerating lookups in the distributed database. We use a Distributed Hash Table (DHT) to store different kinds of ressources in the network. For example, a resource might be a pair <*nickname, current IP address*>. If someone wants to contact another user with a known nickname, the network is queried for the respective ressource, resulting in the IP address the VoIP application must contact. In DHTs each lookup is routed through the network passing several other peers. Each hop of this route adds an additional delay to the overall lookup time. The DHT protocol is responsible for the number of hops that a lookup takes on average. Chord for example finds a ressource within $O(\log N)$ hops in a network with N nodes. The overall lookup time is decreased if either the number of hops is reduced or the network transmission delay is decreased.

In general, connections to geographical close peers have

*Corresponding author: gerald.kunzmann@tum.de

a smaller delay than connections to more distant peers. As geographical positions are available very seldom, proximity is often defined by network transmission delays. [5] shows that abandoning the strict computation of protocol compliant overlay neighbors and instead selecting physically close overlay neighbors reduces the mean lookup duration. In particular, the authors state that, e.g. for the Chord protocol, the mean hop count is still $O(\log N)$ if a node with ID n picks any node in the range $[(n + 2^i); (n + 2^{i+1})]$ as its i^{th} overlay neighbor rather than the exact successor to $n + 2^i$ on the ring.

Simulating proximity requires an accurate network model, where connections have realistic transmission delays. Table 1 gives a short overview of different approaches to model transmission times. The simplest way is to use analytical distribution functions, e.g. negative exponential distributions. While they do not require difficult computations nor huge amount of memory, they are not able to consider the geographical network topology. As a direct consequence different network transmission times between two nodes are calculated for every packet, which also makes high jitter values unavoidable. Thus, using an analytical distribution function is not feasible when simulating proximity-aware protocols.

Storing all inter-node transmission times in a lookup table would lead to very high precision, but is not applicable in huge networks, as the size of the table grows quadratically with the number of nodes.

Modeling the network topology with routers, autonomous systems and links is a common method to build complex models of the internet, and therefore, it is applied by many topology generators as Inet-3.0 [12] or BRITE [1]. The drawbacks of using this method are that it is problematic to acquire real internet topologies and a large amount of memory is required for huge networks. Also, the computation of routing paths and transmission times is complex and therefore slows down each simulation run.

We present a topology model, that is based on network coordinates. It is characterized by a relatively high precision, but low memory and computation costs during the simulation. The required memory scales linear with the number of nodes in the network. The computation of the network coordinates is expensive, but is done offline and the coordinates may be re-used in different simulations. Real Internet measurements are available from CAIDA [2] which allows simulations to be as close as possible to real network conditions. The basic idea is using network coordinates for estimating the transmission time between two nodes. The inter-node transmission time is directly proportional to the geometrical distance in the coordinate space. Note that the inter-node transmission time is not always directly proportional to the geometrical distance in the real world. For example, nodes that are within the same ISP, but are located in different countries or even on different continents, sometimes may be able to communicate with a smaller transmission time than nodes that are in geographical proximity, but belong to different ISPs. However, as shown in Chapterrefresults, a certain correlation between transmission times and geographical proximity is noticeable. In Chapter 2 we describe the 'Global Network Positioning GNP' method that we use to construct the coordinate space. Chapter 3 explains how GNP is used in our simulations and Chapter 4 shows results that are obtained by using this network model before Section 5 finally concludes the paper.

2 Global Network Positioning (GNP)

Global Network Positioning [9] was originally developed for predicting packet delays from one host to another. Each node periodically pings a set of *monitors* (or *landmarks*) and measures the required round trip times (RTT). With this information and the known monitor coordinates, the nodes are able to compute their own position in the geometrical space.

Creating a new d-dimensional coordinate space at first requires calculating the coordinates of the monitors. To achieve a high precision, it is suggested to choose monitors that are as far apart as possible. All round-trip-times between the monitors must be known and the number of monitors n must be greater then the number of dimensions d ($n > d$). The error between the *measured* distance $\hat{t}_{H_1H_2}$ and the *calculated* distance $t_{H_1H_2}$ between the two nodes H_1 and H_2 is defined as:

$$\epsilon(t_{H_1H_2}, \hat{t}_{H_1H_2}) = \left(\frac{t_{H_1H_2} - \hat{t}_{H_1H_2}}{t_{H_1H_2}}\right)^2 \quad (1)$$

The coordinates of the monitors c_{M_i} can then be computed by minimizing the following objective function for every monitor M:

$$f_{obj,M}(c_{M_1}, \ldots, c_{M_N}) = \sum_{M_i, M_j \in \{M_1, \ldots, M_N\} | i > j} \epsilon(t_{M_iM_j}, \hat{t}_{M_iM_j}) \quad (2)$$

After measuring the RTT to at least m ($d + 1 < m \leq n$) monitors, a node can compute its own coordinates c_H by minimizing the following objective function:

$$f_{obj,H}(c_H) = \sum_{M_i \in \{M_1, \ldots, M_N\}} \epsilon(t_{M_iH}, \hat{t}_{M_iH}) \quad (3)$$

The estimated transmission time $t_{H_1H_2}$ between two arbitrary nodes H_1 and H_2 with coordinates $(c_{H_1,1}, \ldots, c_{H_1,d})$ and $(c_{H_2,1}, \ldots, c_{H_2,d})$ can finally be obtained by computing the geometric distance between the

Model	Computation cost	Memory	Comment
Analytical function	simple, inexpensive	$O(1)$	no geographical information high jitter unavoidable
Lookup table	simple, inexpensive	$O(N^2)$	high precision data available
Network topology	complex	high	problematic data acquisition
Coordinates-based	inexpensive, expensive offline comp.	$O(N)$	good precision data available

Table 1. Different approaches for modeling network transmission times

two nodes in the coordinate system:

$$t_{H_1H_2} = \sqrt{(c_{H_1,1} - c_{H_2,1})^2 + \cdots + (c_{H_1,d} - c_{H_2,d})^2} \quad (4)$$

Currently, we are using the Simplex Downhill Method proposed by Nelder and Mead [8] to solve these minimization problems, because it is very easy to implement.

3 Applying GNP for modeling network transmission

We use GNP coordinates in a slightly different way in combination with ping measurements acquired from CAIDA's skitter project [2]. There are 14 monitors available in the dataset (Table 2), that are mostly positioned at DNS roots. These monitors do daily RTT measurements to a list of selected nodes that are spread over the entire IP space. We are not going to use all monitor nodes for the computation of the coordinates, as good values can already be gained with $d+1$ monitors and the computation duration increases significantly if more monitors are used. Using $d = 5$ we achieved nearly accurate transmission times. As mentioned above, it is important to carefully select the monitors. A lot of research has been done in this area [9, 11]. We select our monitors with help of an *maximum separation* algorithm, i.e. we try to select monitors that have a maximized inter-monitor distance (by means of transmission times). This maximization can be solved very easily, as there are only 14 different monitors available, and it leads to good results. Another promising, but more computation expensive method is the *Greedy algorithm*, that chooses the set of monitors that minimizes the average distance error (Equation 1) between all monitors.

Table 3 shows the symmetric RTT matrix achieved from a subset of 6 monitors that we use to build a 5-dimensional coordinate space. The monitor's coordinates can now be calculated by minimizing Equation 2 for all monitors.

The skitter data set comprises no inter-node RTT measurements, but it provides us with RTT measurements from each monitor to about 300.000 hosts (Table 4). Coordinates for these hosts can be computed by minimizing Equation 3 for all hosts. This computationally expensive multi-dimensional minimization problem is solved offline. Coordinates for the Caida dataset have to be computed once, and can then be reused for all simulations, without any further computation costs. The mean transmission time for the Caida measurements is about 80 milliseconds.

Scenarios we are simulating are described in a *source file*, where parameters like number of total participants, number of online nodes and average online times are set. From it, a traffic generator computes all join, leave and search events, as well as the IDs of nodes and content. We call its output *event file*. The event file can then be put into our *coordinates tool*, that assigns a random host from the Caida dataset to each node in the event file. The tool also adds the appropriate coordinates to the event file. Our simulator automatically detects if coordinates are set or not, and uses the coordinates or a negative-exponential distribution to compute transmission times, respectively. Transmission times between nodes are calculated with Equation 4, but would be constant for each transmission between the same two nodes. Therefore, a log-normal distributed jitter is added to the transmission times, if coordinates are used. This proceeding is based on real internet measurements [7] and results in an even more realistic model. A lognormal distribution is denoted as $\lambda(\mu, \sigma^2)$, and its probability density function (PDF) is expressed as:

$$\Phi(x) = \begin{cases} \exp\left(-\frac{1}{2}\left(\frac{\ln(x)-m}{s}\right)^2\right) & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The parameters m and s can be calculated from measurements where the minimum transmission time θ, the mean transmission time μ and the standard deviation σ are known:

$$m = \ln\left(\frac{(\mu-\theta)^2}{\sqrt{\sigma^2 + (\mu-\theta)^2}}\right) \quad (6)$$

Monitor name	Location	IP address
arin	Bethesda, MD, US	192.149.252.8
b-root	Marina del Rey, CA, US	129.9.0.109
cam	Cambridge, UK	128.232.97.8
cdg-rssac	Paris, FR	195.83.250.10
d-root	College Park, MD, US	128.8.7.4
e-root	Moffett Field, CA, US	192.203.230.250
i-root	Stockholm, SE	192.36.144.117
ihug	Auckland, NZ	203.109.157.20
k-peer	Amsterdam, NL	193.0.4.51
k-root	London, UK	195.66.241.155
nrt	Tokyo, JP	209.249.139.254
riesling	San Diego, CA, US	192.172.226.24
uoregon	Eugene, OR, US	128.223.162.38
yto	Ottawa, CA	205.189.33.78

Table 2. CAIDA monitor hosts

	b-root	d-root	i-root	k-root	nrt	ihug
b-root		68.882	186.476	172.536	127.812	185.123
d-root	68.882		118.987	95.266	208.739	229.618
i-root	186.476	118.987		36.523	315.139	319.436
k-root	172.536	95.266	36.523		275.874	312.360
nrt	127.812	208.739	315.139	275.874		138.511
ihug	185.123	229.618	319.436	312.360	138.511	

Table 3. Inter-monitor round trip times (in milliseconds)

	b-root	d-root	i-root	k-root	nrt	ihug
18.166.0.1	84.055	10.535	117.495	85.541	210.628	251.454
81.165.0.1	146.550	85.889	36.159	9.554	284.824	291.408
198.31.255.254	8.777	98.625	177.254	145.013	127.879	196.591
200.63.11.1	249.277	184.413	1060.883	309.182	376.213	523.068
217.200.12.1	172.939	107.576	75.661	27.682	309.860	321.287
...	...	...	...	...	...	...

Table 4. Host-monitor round trip times(in milliseconds)

$$s = \sqrt{\ln\left(\left(\frac{\sigma}{\mu-\theta}\right)^2 + 1\right)} \tag{7}$$

At the moment, we are doing jitter measurements in the Internet to evaluate the parameters $m(\mu)$ and $s(\mu)$ as a function of mean transmission time. We expect narrow lognormal distributions for nodes in close distance and a higher deviation if the foreign node is farther away.

Additionally, our model takes packet loss into account. We assume, that packets are dropped with the same probability. As our model does not construct a detailed physical topology, it is not possible to consider congestion, and therefore higher packet loss rates, in certain regions of the topology.

4 Results

To evaluate the quality of our coordinates, i.e. how exact we can estimate the RTTs between the nodes compared to the real measurements, we use the directional relative error metric:

$$r = \frac{t_{\mathcal{H}_1\mathcal{H}_2} - \hat{t}_{\mathcal{H}_1\mathcal{H}_2}}{\min(t_{\mathcal{H}_1\mathcal{H}_2}, \hat{t}_{\mathcal{H}_1\mathcal{H}_2})} \tag{8}$$

Therefore, we select two monitors, that have not been used to compute the coordinates, and calculate the relative error between them and 2.000 random hosts from our dataset. A directional relative error of plus (minus) one means, that the calculated distance is larger (smaller) by a factor of two as compared to the measured value, whereas a error of zero is a perfect fit. Figure 1 shows the performance of both algorithms. Maximum separation with 6 monitors has a performance which is comparable to the Greedy algorithm with 9 monitors. 81% of the calculated round trip times reveal a relative error of less than 50%. On the other hand, 50% of the calculated round trip times have a relative error of less than 12.3%. We use maximum separation, as it requires significantly less computational effort.

To evaluate the precision of calculated round trip times with respect to the measured times, we have grouped the measured times and the corresponding calculated times in bins of 50 milliseconds and plotted the directional relative error of each pair on a vertical line (Figure 2). The mean directional relative error is indicated by squares, the 25th and 75th percentiles are indicated by the outer whiskers of the line. The figure also shows that GNP performs quite well for distances under 350 milliseconds. A general trend to undershoot in calculated values is apparent; especially for distances of more than 350 milliseconds, GNP undershoots significantly. Still, only 7% of all evaluated distances are more than 350 milliseconds.These large errors result from nodes that are located in areas far apart from the monitor nodes, therefore their coordinates can not be computed precisely.

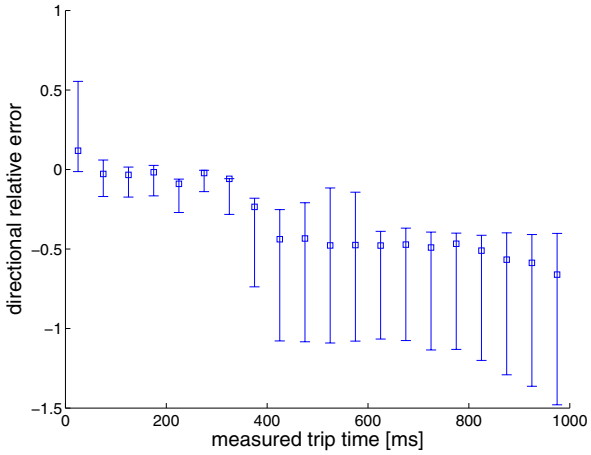
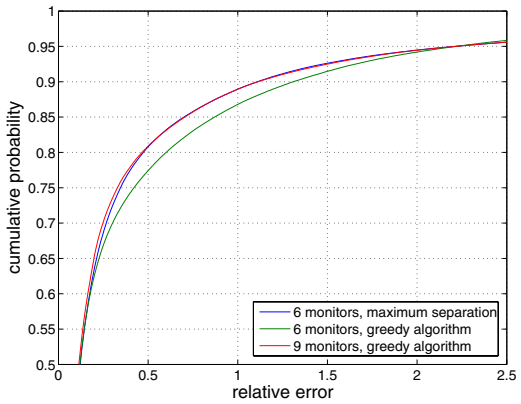


Figure 2. Directional relative error over measured distances

We are mainly interested in using GNP for calculating transmission times for our simulations. Therefore, we compare the distribution of measured trip times from the Caida dataset to trip times calculated with GNP (Figure 3(a)). The average transmission time is the same for both curves! The negative-exponential function has a clearly higher standard deviance ($\sigma = 90.99ms$) than the two other distribution based on realisitic topologies, and there are much more very small ($< 25ms$) and large ($> 200ms$) values.

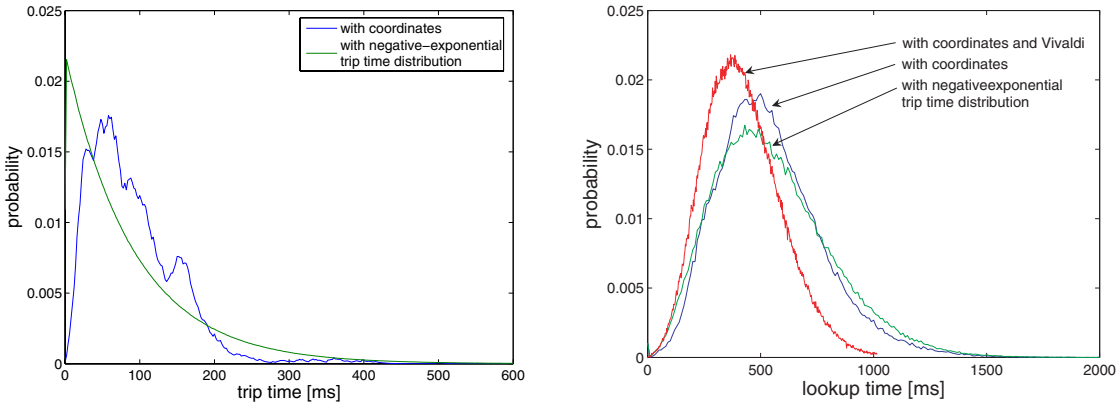
Lookups in DHTs are forwarded through the overlay network, until the responsible node for the queried key is found. This results in a series of packets that are sent over the network, with trip times adding up until the lookup is resolved. The sums of these trip times and small additional local computation delays add up to the total lookup time. Figure 3(b) shows the measured lookup times from simulations with and without using coordinates. As expected, both lookup time distributions are very similar. They look Gaussian, and have approximately the same mean value. The curve corresponding to the negative-exponential distribution is a bit wider, because the standard deviation is bit larger for the negative-exponential distribution. According to the *Central Limit Theorem*, the sum of infinitely many statistically independent random variables has a Gaussian distribution, regardless of the elementary distributions.

Still, the network model based on GNP provides us with a more realistic framework. Therefore, we are able to apply proximity neighbor selection in our finger and search algorithms. Nodes can estimate the transmission times to their neighbors by evaluating existing traffic to this nodes, or by sending active probe packets. Nodes may also predict the distance to another node if network coordinates are applied in the p2p protocol. Network coordinates can be calculated



percentile max. sep.	10	20	30	40	50	60	70	80	90
relative error $\leq$ (in %)	1.83	3.84	6.20	8.90	12.35	17.68	26.93	47.57	111.23

Figure 1. Monitor selection method comparison



(a) Worldmap (b) Calculated coordinates

Figure 3. Trip time distributions (left) and corresponding lookup time distributions (right)

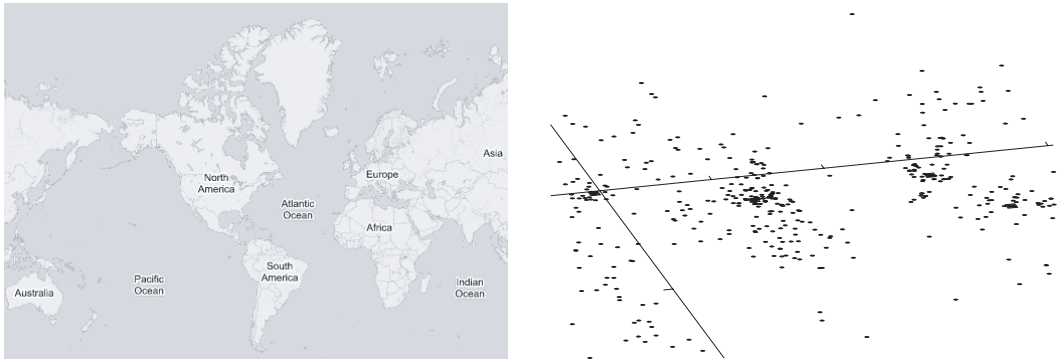
by making use of monitor nodes as it is done with GNP [9] or PCA [10], or by simulating the positions of the nodes with a distributed algorithm like Vivaldi [4, 3].

We are using the Vivaldi coordinates in our VoIP application based on the Chord protocol, as the algorithm is fully distributed and computationally inexpensive. Therefore, it seems particularly suitable for applying it to p2p networks. While the average number of hops is still in $O(\log N)$, the overall lookup time is significantly shorter, because a close node can be selected as next hop of a search request. Figure 3(b) also shows the influence of using a proximity-aware protocol like Vivaldi. The curve is significantly shifted to the left. The average lookup time in this scenario could be reduced to 0.8 times the lookup time using the original Chord protocol.

Another interesting phenomena is shown in Figure 4. If our 5-dimensional coordinates are projected to a 2-dimensional coordinate space, a remarkable amount of clustering can be recognized. If we compare the clusters to a worldmap, even 'continents' may be identified in the coordinates space. This is astoundingly, as coordinates have been calculated from transmission times only. We take this as another fact, that the calculated coordinates are a good representation of the real internet topology.

5 Conclusion

In this paper we presented a scalable and realistic model for internet transmission times. Using a model based on network coordinates enables researchers and developers to simulate large scale (p2p) networks. In a network with N



(a) Worldmap (b) Calculated coordinates

Figure 4. Node distribution in a 2D projection

nodes our coordinate-based model scales with $O(\log N)$, whereas a simple lookup table is of size N^2. Its main advantage compared to other models (using analytical functions) is the fact, that the transmission delay between any two nodes is not random (based on, e.g., a negative-exponential distribution) but is constant over time. An additional jitter (using a log-normal distribution) and a constant packet loss rate make our model even more realistic. This feature is mandatory to simulate proximity-aware protocols. We showed how to calculate network coordinates from a given dataset of transmission times by minimizing the relative error between measured and calculated distances between the peers. We also evaluated the approximation error and precision of this model. Finally, we demonstrated how to apply our approach to a network simulator. We also discussed how to use this realistic model to develop a protocol that exploits proximity. Thus, we could reduce the overall lookup time by 20% in our VoIP scenario.

This paper is part of our current work on the efficient simulation of large-scale p2p networks. The corresponding technical report [6] addresses many other topics like wether to simulate on packet or on application level, how to model bandwidth in filesharing systems or how to design efficient data structures and event algorithms.

Acknowledgments

The authors would like to thank Prof. Jörg Eberspächer, Prof. Phuoc Tran-Gia, and Prof. Ulrich Killat for enabling and supporting this work.

References

[1] Brite: Boston university representative internet topology generator. `http://www.cs.bu.edu/brite/index.html`.

[2] Cooperative association for internet data analysis (CAIDA). `http://www.caida.org`.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, OR, USA, August 2004.

[4] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, USA, March 2004.

[5] R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. 2003.

[6] T. Hoßfeld, A. Binzenhoefer, D. Schlosser, K. Eger, J. Oberender, I. Dedinski, and G. Kunzmann. Towards efficient simulation of large scale p2p networks. Technical Report 371, University of Wuerzburg, 2005.

[7] T. Hoßfeld, A. Mäder, K. Tutschku, P. Tran-Gia, F.-U. Andersen, H. de Meer, and I. Dedinski. Comparison of Crawling Strategies for an Optimized Mobile P2P Architecture. Technical Report 356, University of Würzburg, 4 2005.

[8] J. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[9] T. E. Ng and H. Zhang. Towards global network positioning. In *Internet Measurement Workshop, Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement 2001*, pages 25–29, San Francisco, CA, US, November 2001.

[10] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Internet Measurement Conference, Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement 2003*, pages 143–152, Miami Beach, FL, USA, October 2003.

[11] L. Tang and M. Crovella. Geometric exploration of the landmark selection problem. In *Lecture Notes in Computer Science 3015, Proceedings of Passive and Active Measurement Workshop (PAM2004)*, pages 63–72, Juan-les-Pins, FR, April 2004.

[12] J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical Report CSE-TR-456-02, Department of EECS, University of Michigan Ann Arbor, 2002.