A Simulation Framework for Distributed Super-Peer Topology Construction Using Network Coordinates

Peter Merz Matthias Priebe Steffen Wolf
Distributed Algorithms Group
University of Kaiserslautern
Kaiserslautern, Germany
{pmerz,priebe,wolf}@informatik.uni-kl.de

Abstract

Scalability constitutes a property of particular interest when investigating Peer-to-Peer overlay behavior. Recent advances that improve scalability include super-peer infrastructures and network coordinates. To this end, we introduce a lightweight simulation framework that focuses on Peer-to-Peer topology construction supported by network coordinates and optionally, super-peers. One of the design goals has been a straightforward way to re-use simulation code in practical applications. We illustrate the utility of our simulation framework by means of a comprehensive application case study that compares different network coordinates generation mechanisms in the context of a distributed super-peer topology construction effort.

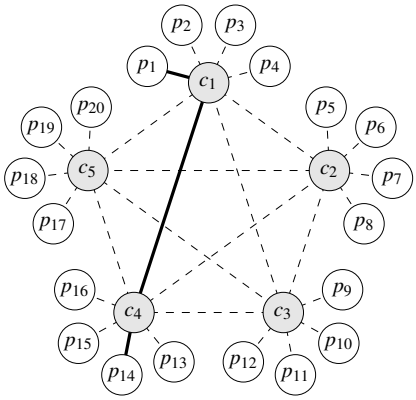

Figure 1. Example of a P2P network with selected Super Peers

1 Introduction

A number of simulators for Peer-to-Peer (P2P) overlay networks have been proposed. To address a large number of needs and demands, these simulators exhibit a broad range of individual properties, however researchers occasionally still conclude that no existing simulator properly fits their requirements [19]. We have developed a lightweight simulation framework, called *NetSim*, that deliberately focuses on the discrete event-based simulation of P2P topology construction with the aid of network coordinates. NetSim has been designed to be convertible into a real distributed application, implementing a previously simulated protocol, with comparatively little effort, meaning that simulation results may be verified in a real setting. The benefit of re-using code in this way has been pointed out in [19].

Well-known properties of fully decentralized P2P systems include self-organizing and fault-tolerant behavior. However, the scalability of such networks becomes an issue in the case of substantial growth. For example, the Gnutella P2P network in its original version exhibits properties that hamper scalability in large-scale settings [27]. In such cases, inefficient communication schemes prevent the network from growing smoothly [17, 5]. A feasible solution to this issue is the introduction of *super-peers*. Super-peers are peers that act as servers for a number of attached common peers, while at the same time, they form a network of equals among themselves. That way, the entire P2P network may remain connected over a considerably smaller number of links than before [27, 5, 10]. An example P2P application that uses a super-peer topology is the *FastTrack/KaZaA* filesharing network [13].

Generally, we strive for a topology in which a subset of the nodes will function as super-peers while the rest of the nodes, henceforth called edge peers, is each assigned to exactly one of the super-peers, respectively. The super-peers are fully connected among themselves. They are able to communicate directly with the edge peers assigned to them and with their fellow super-peers. The edge peers, however, will need to route any communication via their assigned super-peer. For the scope of this paper, the set of super-peers C is referred to as *core*. Figure 1 depicts an example

0-7695-3089-3/08 $25.00 © 2008 IEEE
DOI 10.1109/PDP.2008.47

of a super-peer topology with five super-peers.

Super-peer overlay networks do not classify as structured networks like DHT-based overlays which are built on distributed hash tables [13, 6], but do not exhibit a completely unstructured fabric either as they incorporate a hierarchical structure of peers [27]. However, a DHT-based overlay can easily be incorporated into the core as a replacement for the fully meshed network, reducing the number of connections for the super-peers.

The distributed construction of a super-peer topology may require frequent distance (delay) measurements between nodes. Network coordinates effectively tackle this issue. These are space coordinates individually assigned to every node in such a way that the spatial distance between any two nodes approximates the true network delay between those nodes.

In the upcoming sections, we present NetSim and demonstrate its utility in the context of an application example which compares the effectiveness of various coordinates-generation mechanisms for the purpose of distributed super-peer topology construction.

The remainder of this paper is organized as follows. In Section 2, we introduce our simulation framework. In Section 3, we present a protocol for super-peer topology construction and maintenance, including a distributed algorithm that builds super-peer topologies. In Section 4, we discuss network coordinates mechanisms. In Section 5, we define the scenario of our experiments. In Section 6, we present and compare the experiments' results. In Section 7, we provide an overview of related work. The paper concludes with an outline for future research in Section 8.

2 Simulation Framework

NetSim is a lightweight, event-based simulation framework, dedicated to the simulation of P2P overlay construction and maintenance. It concentrates on the overlay itself and generally abstracts from lower layers except for the message delay (round-trip time, RTT) between two interconnected nodes. By construction, it requires RTT information to be generally available for node-to-node communication delays.

NetSim has been developed in the Java language for portability reasons, featuring a modular, object-oriented structure. Besides a concentration on the overlay level and the integration of network coordinates, a major design goal has been the provision of a simple, straightforward way of transforming the implemented protocol into a real application that performs actual work in a distributed fashion.

NetSim is based on the message concept, hence all communication is asynchronous. Nodes in the overlay exchange messages and act upon reception of a message. The reception of a message generally affects the recipient's local state, possibly resulting in the creation of additional messages sent to other nodes.

While NetSim can handle arbitrary P2P topologies, we focus on super-peer overlays for scalability reasons, and use NetSim for the simulation-based evaluation of such overlays. A protocol is implemented by inheriting from given generic Java classes that model individual node behavior such as sending messages and joining or leaving the overlay. NetSim features a number of overlay supervision functions, including a snapshot facility that may be called in regular intervals to perform arbitrary tasks with global view.

NetSim centers around a message queue which distributes messages to individual nodes. Messages contain their expected time of arrival (ETA) based on the communication delay. When a node sends a message, the message will be placed in the queue according to its ETA. This event-based simulation proceeds in time in a stepwise manner, removing the foremost message from the queue which is due to be the next message received by a node in the overlay. Also, nodes may trigger local events by placing messages in the queue that are addressed to themselves. These messages constitute a special kind of queue element; in particular, their ETA is set to the desired point in time when the local event should be triggered. Simulation stops when a predefined point in time or a desired event has been reached.

NetSim's design goal of being easily transformable into a real distributed application has influenced its mechanisms. Once a protocol has been implemented with NetSim and simulation yields promising results, NetSim's message queue may be removed, and send/receive operations may be replaced by counterparts which perform actual message handling. Local events may be handled by timer-triggered callbacks. Along with a bootstrapping mechanism, this is already sufficient to make the implemented protocol work practically in a distributed environment.

Bootstrapping is generally handled by having a new node contact an arbitrary node which already belongs to the overlay network. Actual node behavior and subsequent actions depend on the respective protocol implementation.

NetSim integrates facilities to deal with two different kinds of node-to-node delays. One kind is the actual round-trip time (RTT) between any two nodes, given by a squared matrix read from a file. Another kind of delay taken into account by NetSim is a RTT estimate provided by network coordinates that are established at simulation run-time. The simulated protocol is supposed to generally rely on the RTT estimate, while the actual RTT for a link will be available to nodes adjacent to that link only. We refer to the special case that the actual RTT is used as the RTT estimate for all node-to-node distance inquiries as *flawless embedding* which causes no network coordinates-related error but incurs excessive network load when exercised in practice.

If scalability in simulation becomes a significant issue,

NetSim can also use given network coordinates instead of a distance matrix. This will reduce the accuracy of the simulation as triangle inequality violations or missing links cannot occur in a pure network coordinates approach. Moreover, the modularity of NetSim allows more complex topologies to be used. However, in the topology construction example presented in this paper, the accuracy of the round-trip-times is of utmost importance, hence a distinct distance matrix needs to be deployed.

General criteria for the evaluation and differentiation of P2P simulators were presented in [19, 18]. NetSim fits into this scheme as follows:

- *Simulation architecture:* NetSim is event-based, designed for both structured and unstructured P2P overlay networks, and possibly enhanced by super-peers, with a special focus on the integration of network coordinates.
- *Usability:* NetSim is based on the Java platform, with no dedicated scripting language. Simulation parameters are defined in a configuration file.
- *Scalability:* Constrained by the limitations of RTT data integration, we have strived to foster scalability based on NetSim's lightweight approach to simulation.
- *Statistics:* NetSim provides both general, event-based and protocol-dependent statistics from an individual peer's point of view. A snapshot mechanism provides a global view on a regular basis for arbitrary purposes.
- *Underlying network:* Discarding details from lower layers except for the communication delays, NetSim concentrates on the overlay itself.

In Section 7, NetSim is compared to a number of P2P simulators.

3 Topology construction and maintenance

In this section, we describe a protocol that builds and maintains a P2P topology augmented by a super-peer infrastructure. For the purpose of selecting nodes as super-peers, the protocol resorts to a distributed algorithm which we will present subsequently.

3.1 Overlay maintenance protocol

We have designed and implemented a protocol for super-peer overlay network construction and maintenance.

In the bootstrapping phase, a node X that wants to join the overlay first contacts an arbitrary node Y known to already participate in the overlay, and sends a Q S - P message to Y. Y responds with a list of all super-peers it knows, encapsulated in a S P L message. X picks the closest super-peer Z from the list and asks Z for being accepted as one of its edge peers, sending a C - message to Z. Y and Z may be identical. Depending on its local state, Z replies with either A or R . In the case of A , X is now linked to Z as one of its edge peers, and as such, X has been accepted as a participant in the overlay, while in the case of R , X will locally mark Z as unavailable, pick another super-peer from its list and try again. When X wants to disconnect from Z later, X sends a D message to Z, which requires no reply.

A super-peer K may pick one of its edge peers, L, and appoint it super-peer. In this case, K first selects a suitable edge peer L, then sends a N S P message to L. If L accepts the offer, it replies with an A S P message. Eventually, L broadcasts its presence by sending I S P messages to all super-peers.

Conversely, a super-peer K may find itself wishing to downgrade to edge peer level and redistribute its currently supported set of edge peers to other super-peers. A similar situation may arise if K determines one of its edge peers, L, to be a better super-peer than itself. In the latter case, K sends a R message to all of its edge peers excluding L, asking them to switch to L voluntarily, and waits a pre-defined time span, labeled as the *Pre-Reject Phase*, while in the former case, a similar message is sent without containing a redirection to a particular super-peer. Both cases converge at this point. K now sends R messages to each of its remaining edge peers, disconnecting them. K also sends R S P messages to all super-peers which in turn may forward this information to their respective edge peers.

A peer changes its super-peer by sending a C message to the new super-peer and, in case it is accepted, a D - message to the old super-peer. The super-peer responds with an A or R message.

The predominant amount of messages is expected to be generated by the insertion or degradation of a super-peer, since this information has to be forwarded to all nodes, while all other messages are only exchanged between one edge peer and its super-peer. Super-peers keep track of super-peer memberships and provide common peers with lists of super-peers, particularly in the case they issue a R - message to a prospective edge peer. This procedure may be backed up by a gossip mechanism that provides super-peer list synchronization in the core.

3.2 Super-Peer Selection Algorithm

We present a distributed Super-Peer Selection Algorithm (SPSA) as an integral part to the aforementioned protocol, aiming to minimize the total communication cost. In a P2P setting, this cost can be thought of as the total all-pairs end-to-end communication delay. SPSA aims for $|C| = \sqrt{n}$ super-peers and $\sqrt{n} - 1$ edge peers per super-peer, with n being the total number of peers in the overlay. This value minimizes the number of links each super-peer has to maintain. Increasing the number of super-peers also increases the number of intracore connections, while decreasing the

```
every γ time units do:
  i ← nearestSuperPeer()
  if mysuperpeer ≠ i then
    mysuperpeer ← i
    connectTo(i)
  end if
```

Figure 2. SPSA, edge peer part

```
every γ time units do:
  if |E_k| < 1/2 · |C| then {downgrade}
    role_k ← edge-peer
  else if |E_k| > 2 · |C| then {promotion}
    j ← most suitable edge peer
    role_j ← super-peer
  else {replacement}
    j ← most suitable edge peer
    if replacement with j yields gain then
      request j to become super-peer
      if j accepts, downgrade to edge peer level
    end if
  end if
```

Figure 3. SPSA, super-peer part for a super-peer k

number of super-peers increases the number of spokes, in both cases outweighing the respective savings.

SPSA is meant to run on every peer in the overlay. With SPSA, a peer regularly performs a reorganization cycle every γ time units. In the reorganization cycle, an edge peer changes its super-peer if it finds another super-peer to be closer, as shown in Fig. 2. Should the connection to its super-peer be lost, an edge peer restores its link to the overlay core by contacting the closest of the remaining super-peers from its list. A super-peer either stays put or performs a reorganization step.

The super-peer reorganization step consists of a choice among several options, always based on the current length of the super-peer list $|C|$. A super-peer relinquishes its role, downgrading to edge peer level, if it finds the number of attached edge peers, h, too low ($h < 1/2 \cdot |C|$). Conversely, if h exceeds the maximum load threshold ($h > 2 \cdot |C|$), the super-peer picks one of its edge peers and promotes it to super-peer level. A third reorganization option besides downgrade and promotion is replacement. A super-peer replaces itself with one of its edge peers, downgrading to edge peer level while promoting one of its edge peers to become super-peer, and relocates its current edge peers to the new super-peer, if it finds that particular edge peer to be able to deliver the super-peer's service at lower cost. The super-peer reorganization cycle is depicted in Fig. 3.

SPSA has been designed to operate in a distributed way. There is no global view required, and the algorithm fits seamlessly into the super-peer topology construction and maintenance protocol. NetSim is able to handle churn. However, for the course of this publication, we have neglected churn with regard to SPSA for reasons of scope.

4 Network Coordinates

Network coordinates are space coordinates individually assigned to each node in a given network such that the distance between any two nodes in that space approximates the measurable delay between those nodes in the network. Once a node has learned of another node's coordinates, it may estimate the delay to the remote node by evaluating the spatial distance at virtually no cost instead of sending delay measurement packets (*ping* packets) over the network. Delay estimation using network coordinates intends to cause only a fraction of the network load which would be incurred with standard delay measurement methods [22, 3, 9, 20].

Network coordinates mechanisms can be divided into two categories. Static approaches compute coordinates for every node only once. There is no explicit need to recompute a node's coordinates. In contrast, dynamic methods are based on the recomputation of nodes' coordinates over time. This contrast also affects the message complexity as dynamic methods require an ongoing message exchange to update coordinates while static methods do not.

4.1 Vivaldi

A popular example for a dynamic method is the *Vivaldi* system which relates to the effect of spring deformation [4]. It can be initialized by providing arbitrary initial coordinates to each participating node. When two nodes communicate, they measure both their link's delay and their spatial distance, then set up a virtual spring. Should measured delay and spatial distance be identical, the spring rests in a zero-energy state that represents the optimum. Otherwise, the spring is relaxed or compressed, applying a force to the nodes which either pulls them closer toward each other or pushes them apart in the space. This way, the energy contained in the spring reflects the mismatch between measured delay and spatial distance. The system strives to reduce that energy for all springs to a minimum, preferably zero, level.

4.2 GNP

In contrast, a widespread static method is *General Network Positioning* (GNP) [20] that picks particular nodes to act as landmarks which serve as points of reference for other nodes that wish to embed themselves into the coordinate space. This approach is implemented using an optimization-driven scheme in which nodes solve optimization problems to embed themselves into the target space.

Several properties of GNP have been investigated and improved by subsequent work. GNP's original requirement

of fully fixed landmarks has been abolished through an effort brought forward in [3], and an advanced method to solve a major part of the optimization problems has been introduced in [16]. We use both aforementioned improvements and refer to the improved GNP as *GNP**.

4.3 Geographical coordinates

Distance estimation based on geographical coordinates is a special kind of a static mechanism. While both Vivaldi and GNP are based on latency-related measurements, implicitly incorporating link capacity and utilization, a geographical approach assumes the distance of any two nodes to strongly correlate with their round-trip time, focusing on location only. Concisely, the distance between any two points on the Earth's surface with known latitude and longitude equals a great circle's section running through both points. The authors of GNP have examined the aspect of geographical coordinates in their work [20] and found GNP's distance measurement to go beyond mere geographical relationships. We have included geographical coordinates as a straightforward means to provide delay estimates.

5 Simulation Set-Up

We have set up a series of experiments to compare all coordinate-generation mechanisms that have been introduced in Section 4 by means of simulation using NetSim. For these experiments, we used real-world node-to-node delay information from PlanetLab [1], a world-wide platform for performing Internet measurements and distributed computing. Jitter information was unavailable. Its inclusion would have caused issues regarding the estimation of the network coordinates embedding error and SPSA's objective function. From the measurements reported in [1] we used the first measurement for each month in 2005 (denoted by mm-2005), and also retrieved the respective IP addresses. Those networks consisted of 70 to 419 nodes. We based our simulations on the round-trip times (RTT) from any host to any other host as the communication cost for the edges in the overlay network. For each experiment, resulting values were averages over 30 runs.

For benchmark purposes, we used an objective function Z that returns the total communication cost for all participating nodes, when all communication is routed over the given super-peer configuration. Its evaluation temporarily required global view, a requirement natively covered by NetSim's snapshot facility. Formally, Z may be written as

$$Z = \sum_{i} \sum_{j:j\neq i} \left(d_{i,s(i)} + d_{s(i),s(j)} + d_{s(j),j} \right)$$

where $d_{i,s(i)}$ expresses the distance (cost in terms of true round-trip time as specified by the input matrix) between peer i and its super-peer $s(i)$, term $d_{s(i),s(j)}$ the distance between the super-peers for i and j, and $d_{s(j),j}$ the distance between peer j and its super-peer. If either node i or j is currently serving as super-peer, the corresponding distance $d_{i,s(i)}$ or $d_{s(j),j}$ is zero.

NetSim has successfully handled the largest networks for which RTT information was publicly available, including a 2,500-node network provided by the authors of Meridian [26], but IP addresses and therefore geographical coordinates were unavailable for those networks' nodes, hence we have focused our investigation on PlanetLab data.

We allowed for flawed matrices with some missing delay entries; in such cases, NetSim computed a two-hop distance via a random node to the destination as a replacement. All PlanetLab RTT matrices contained flaws of that kind, which were successfully treated. In practice, distance measurement mechanisms should return the minimum or median of a sufficiently large sample.

In addition, we have acquired a mapping from the IP address space to the space of geographical coordinates from a public source. We have refrained from resorting to the *NetGeo* project used in [20] as it has stopped active maintenance of its data years ago. Apart from being up-to-date, we required the database to be a stable release (no beta status) which is freely available to the general public with no restrictions on the maximum number of queries. Additional requirements included coverage of the entire IP address space, a known maintainer who supervises the quality of coordinates information, and no need for extra infrastructure to be deployed. All requirements were fulfilled by GeoLite City from MaxMind LLC [15]. Using GeoLite City, we have assigned longitude and latitude information to known PlanetLab node IP addresses, allowing for geographical coordinates to provide a distance measure.

We expect simulation results to be influenced by two error sources in particular. The first source is the error introduced by network coordinates, which splits itself again into an embedding error stemming primarily from violations of the triangle inequality [14, 28] and a gap between the optimum and the achieved solution for heuristical approaches to computing such coordinates. The second error source, introduced by the distributed algorithm, is the gap between the optimum and the proposed super-peer selection.

Each simulation run lasted for 900 000 time units, where one time unit corresponded to one millisecond of simulated real time. Reorganization cycles were scheduled individually for each node every $\gamma = 4000 \pm 20$ time units. This value was chosen to strike a balance between low message complexity and swift reaction to topology changes. In the beginning, one random peer was chosen to act as the first super-peer. Peers were not synchronized.

With regard to network coordinates, we used both Vivaldi and GNP*. With Vivaldi, we have initially placed the nodes at positions drawn from a uniform random distribution, where each coordinate was contained in the range

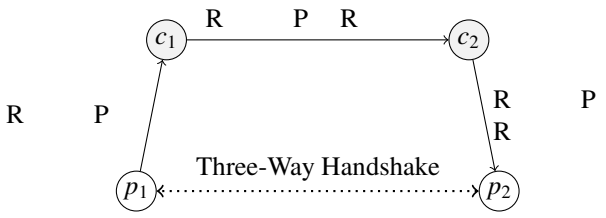

Figure 4. Picking a remote peer for Vivaldi's RTT estimation

[0, 500], such that in a 4-dimensional space, the initial distance between any two nodes did not exceed 1000. In line with the Vivaldi mechanism, nodes were free to push themselves farther away from each other as required. Nodes sent requests for distance estimation to other peers every 2000 time units, routed to random destinations via the super-peers, which then established their pairwise force using a three-way handshake. Figure 4 depicts an example where peer p_1 issues a Ping request to its super-peer, c_1. c_1 randomly decides to forward the request to another super-peer, c_2, which is also chosen randomly. c_2, again, randomly decides to forward the request to one of its edge peers, p_2. When the request arrives at p_2, it promptly responds to the request, contacting p_1. With GNP*, static coordinates were computed at the experiment's start with no subsequent update and spread to other peers using the common piggybacking strategy. When routing involves an intermediate hop between two super-peers, this hop enables the transmitting super-peer to transfer its coordinates to the receiving super-peer, ensuring the dissemination of dynamic super-peer coordinates.

The simulation covered major aspects of self-organizing P2P behavior as edge peer connections to super-peers were established, rejected, or dissolved. Nodes in our event-based simulation were restricted to a local view of the network. Every node was equipped with a possibly outdated super-peer list. We abstracted from the actual node coordinates exchange and assumed that edge peers received super-peers' coordinates via their assigned super-peer, while super-peers knew all remaining super-peers' coordinates, and super-peers knew their attached edge peers' coordinates. In practice, this is achieved by having every node piggyback its coordinates on all outbound messages.

6 Results

This section provides a breakdown of our experiments' outcome. Table 1 contains the final objective function values representing total all-pairs communication cost for all 12 tested PlanetLab networks, normalized to the outcome of True RTT. The table compares, for every network, the performance of the distributed super-peer selection algorithm embedded in the super-peer topology construction and maintenance protocol using various kinds of coordinates mechanisms (geographical, Vivaldi, GNP*, and True RTT reflecting a flawless embedding). It also gives the normalized objective function value for a random assignment of super-peers as a benchmark.

Network	Size	Geo	Vivaldi	GNP*	Random
01-2005	127	1.61	1.37	1.42	3.40
02-2005	321	1.50	1.25	1.41	2.94
03-2005	324	1.61	1.21	1.44	3.13
04-2005	70	2.05	1.15	1.29	2.55
05-2005	374	1.50	1.21	1.36	2.83
06-2005	365	1.89	1.38	1.58	3.08
07-2005	380	1.41	1.30	1.27	3.03
08-2005	402	1.64	1.38	1.36	3.29
09-2005	419	1.65	1.41	1.41	3.09
10-2005	414	2.66	1.51	1.56	3.88
11-2005	407	1.81	1.38	1.52	3.09
12-2005	414	2.67	2.03	2.22	6.23

Table 1. Objective function values for all 12 tested networks, normalized to the True RTT outcome

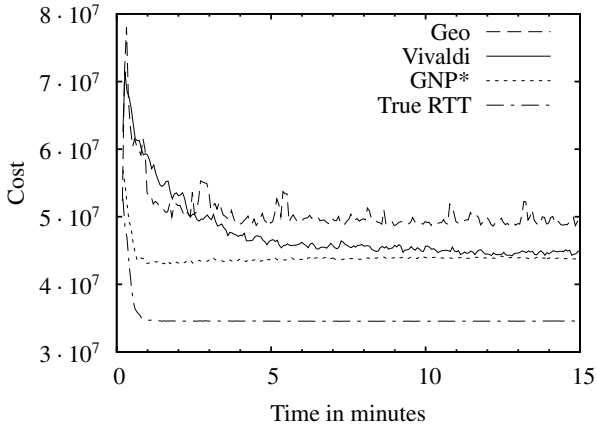

Figure 5. Objective function value for 07-2005

Fig. 5 depicts the objective function values as *Cost* plotted over time for the 07-2005 network.

Judging from these results, Vivaldi fares best in most experiments, but needs some time for set-up and continues to slightly oscillate. With both GNP* and Vivaldi, the desired number of $\sqrt{n}$ super-peers is closely approximated or reached. GNP* quickly creates stable coordinates and performs only marginally worse than Vivaldi in our experiments. The gap between a flawless embedding's performance (i. e. network coordinates mechanisms generate no error) and the outcome delivered by the latency-based mechanisms suggest the impact of triangle inequality vio-

lations hampering the network coordinates-based distance estimation, however a flawless embedding represents only a theoretical lower bound as a full probing of the network would be required, an effort that network coordinates explicitly wish to avoid [14].

However, geographical coordinates lead to the creation of substantially worse topologies when rated by objective function value. We find the reason to be twofold. First, individual link capacity and utilization are not incorporated into distance estimation, as suggested in [20]. Moreover, the estimate's quality also depends on the underlying database. In the case of GeoLite City, we found that Chinese PlanetLab nodes were all mapped to a single location in Beijing, China. This lack of resolution distorts distance estimates.

With all coordinate-generating mechanisms, convergence has been achieved before the simulated period of time elapsed.

7 Related Work

A recent survey related to P2P research revealed a number of simulators currently used [19]. The survey's authors found none of these simulators to fit all requirements. 8 out of 9 major P2P overlay simulators investigated were based on the Java platform. All were free software, however different kinds of license policies applied.

When compared to other event-based P2P simulators that concentrate on the actual overlay rather than also considering the layers below, NetSim proves to be different. *OverSim* is a P2P simulator based on OMNeT++ with a focus on scalability [2]. As with NetSim, code reuse for practical application has been a design goal. However, OverSim does not fully address our requirements as the modelling of node-to-node delays is too simplistic; it is based on either a constant value or placing the nodes in a two-dimensional space. Simulators limited to structured overlays include *P2PSim* [11] and *OverlayWeaver* [18, 2]. Like NetSim, *PlanetSim* and *PeerSim* support both structured and unstructured overlays and both run on the Java platform, however they have other shortcomings with regard to our requirements: PlanetSim offers no support for gathering statistics, while PeerSim focuses on epidemic algorithms, which is at best a side aspect of our research, and disregards latencies [18]. In [7], delays between hosts were modelled using network coordinates to circumvent the need for node-to-node delay matrices. This approach permits the consideration of large numbers of nodes but prevents the estimation of embedding errors due to lack of actual measurements. NetSim differs from other simulators in the emphasis that it puts on coordinates-based delay estimation and the requirement for a node-to-node delay input matrix.

Previous proposals for super-peer topology construction mechanisms include *SG-1* [17], a gossip-based protocol which interconnects the super-peers with a random structure. SG-1 aims for a minimum number of super-peers according to given capacity constraints. Its successor, *SG-2* [5], also operates on the basis of a gossip mechanism, however SG-1 optimizes for available bandwidth while SG-2 strives to minimize communcation cost in terms of latency. Our protocol differs in a number of key points from SG-1 and SG-2. Because of the fully meshed core, messages are quickly propagated through the overlay. Also, our protocol handles super-peer capacity with implicit thresholds rather than explicit capacity limits.

There are several approaches to computing latency-based network coordinates. Vivaldi and GNP are two widely accepted ones. *Big-Bang simulation* is a dynamic approach similar to Vivaldi [24]. In the field of static approaches, one implementation of the scheme apart from GNP is the combination of Lipschitz embedding, which uses delays to landmarks as coordinates, and Principal Component Analysis to reduce the Lipschitz embedding's dimensionality [12, 25]. An alternative landmark-based approach uses virtual bins to group nodes which have the same delay order with regard to the landmarks [23]. However, with the binning approach, there is no direct way to determine quantitative distance estimates.

With respect to geographical coordinates for Internet nodes, we have considered a number of alternatives to NetGeo as used by [20]. In [8], geographical properties of the Internet were investigated, and the two mechanisms deployed there, Ixia's *IxMapper* and Akamai's *EdgeScape*, offer no open access but are available on a commercial basis only. An additional three methods for mapping IP addresses to geographical coordinates, *GeoTrack*, *GeoPing*, and *GeoCluster*, were introduced in [21], but neither has been publicly deployed. Generally, an investigation into the quality of databases that map IP addresses to geographical sites might discover issues pertaining to a lack of resolution in several areas around the globe.

8 Conclusion

We have introduced NetSim, a lightweight, event-based simulation framework for P2P overlay construction and maintenance that gives special attention to network coordinates, yet does not neglect the aspect of scalability. NetSim provides the opportunity to re-use vast parts of the code as implemented protocols may be easily transformed into distributed real-world applications with only minor modifications to the surrounding framework.

We have presented a protocol for distributed super-peer topology construction and maintenance, including a decentralized algorithm picking suitable nodes to serve as super-peers. In this context, we have also conducted a series of experiments to compare the effectiveness of various network coordinates mechanisms supporting the super-peer topology construction effort, including geographical coordinates.

Future work includes the extension of the protocol to handle churn and message loss, and the deployment of middleware for distributed computing based on NetSim.

References

[1] S. Banerjee, T. G. Griffin, and M. Pias. The Interdomain Connectivity of PlanetLab Nodes. In C. Barakat and I. Pratt, editors, *Proceedings of the 5th International Workshop on Passive and Active Network Measurement*, pages 73–82, 2004.

[2] I. Baumgart, B. Heep, and S. Krause. OverSim: A Flexible Overlay Network Simulation Framework. In *Proceedings of the Tenth IEEE Global Internet Symposium (GI'07) in conjunction with IEEE INFOCOM 2007*, Anchorage, USA, May 2007.

[3] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for distance estimation. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, pages 178–187, 2004.

[4] R. Cox, F. Dabek, M. F. Kaashoek, J. Li, and R. Morris. Practical, distributed network coordinates. *Computer Communication Review*, 34(1):113–118, 2004.

[5] G. P. Jesi, A. Montresor, and Ö. Babaoglu. Proximity-Aware Superpeer Overlay Topologies. In A. Keller and J.-P. Martin-Flatin, editors, *Proceedings of SelfMan'06*, volume 3996 of *Lecture Notes in Computer Science*, pages 43–57. Springer, June 2006.

[6] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *STOC'97: Proceedings of the 29th annual ACM symposium on Theory of computing*, pages 654–663, New York, USA, 1997. ACM Press.

[7] G. Kunzmann, R. Nagel, T. Hoßfeld, A. Binzenhöfer, and K. Eger. Efficient simulation of large-scale P2P networks: Modeling network transmission times. In *Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'07)*, pages 475–481, 2007.

[8] A. Lakhina, J. W. Byers, M. Crovella, and I. Matta. On the geographic location of Internet resources. In *Proceedings of the 2nd Usenix/ACM SIGCOMM Internet Measurement Workshop 2002, Marseille, France*, pages 249–250, 2002.

[9] J. Ledlie, P. R. Pietzuch, and M. I. Seltzer. Stable and accurate network coordinates. In *Proceedings of the 26th International Conference on Distributed Computing Systems*, 2006.

[10] D. Li, N. Xiao, and X. Lu. Topology and resource discovery in peer-to-peer overlay networks. In *Grid and Cooperative Computing – GCC 2004 Workshops*, volume 3252 of *Lecture Notes in Computer Science*, pages 221–228, 2004.

[11] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In *Proceedings of INFOCOM*, pages 225–236, 2005.

[12] H. Lim, J. C. Hou, and C.-H. Choi. Constructing Internet coordinate system based on delay measurement. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 129–142, 2003.

[13] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93, 2005.

[14] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the accuracy of embeddings for Internet coordinate systems. In *Proceedings of the Internet Measurement Conference*, 2005.

[15] MaxMind. GeoLite City. Website. `http://www.maxmind.com/app/geolitecity`.

[16] P. Merz and M. Priebe. A New Iterative Method to Improve Network Coordinates-Based Internet Distance Estimation. In *Proceedings of ISPDC 2007 – 6th International Symposium on Parallel and Distributed Computing*, pages 169–176, Hagenberg, Austria, 2007.

[17] A. Montresor. A robust protocol for building superpeer overlay topologies. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, pages 202–209, 2004.

[18] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai. A Survey of Peer-to-Peer Network Simulators. In *Proceedings of The 7th Annual Postgraduate Symposium*, Liverpool, UK, 2006.

[19] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *Computer Communication Review*, 37(2):95–98, 2007.

[20] T. S. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM*, June 2002.

[21] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 173–185, New York, USA, 2001. ACM Press.

[22] P. R. Pietzuch, J. Ledlie, M. Mitzenmacher, and M. I. Seltzer. Network-Aware Overlays with Network Coordinates. In *Proceedings of the Workshops of the 26th International Conference on Distributed Computing Systems*. IEEE Computer Society, 2006.

[23] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of INFOCOM*, pages 1190–1199, 2002.

[24] Y. Shavitt and T. Tankel. Big-bang simulation for embedding network distances in Euclidean space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, 2004.

[25] L. Tang and M. Crovella. Virtual landmarks for the Internet. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 143–152, New York, USA, 2003. ACM Press.

[26] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proceedings of SIGCOMM*, pages 85–96, 2005.

[27] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering*, pages 49–62, 2003.

[28] H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet Routing Policies and Round-Trip-Times. In *Proceedings of the Passive and Active Network Measurement Workshop*, pages 236–250, 2005.