On the Construction of a Super-Peer Topology underneath Middleware for Distributed Computing

Peter Merz
Distributed Algorithms Group
University of Kaiserslautern
Kaiserslautern, Germany

Jan Ubben
Distributed Algorithms Group
University of Kaiserslautern
Kaiserslautern, Germany

Matthias Priebe
Distributed Algorithms Group
University of Kaiserslautern
Kaiserslautern, Germany

E-mail: {pmerz,j_ubben,priebe}@informatik.uni-kl.de

Abstract

Scalability constitutes a key property of Peer-to-Peer overlay networks. Recent advances that improve scalability include super-peer infrastructures and network coordinates. As an integral part of an upcoming middleware for distributed computing, we have developed a mechanism that builds and maintains a super-peer overlay network in a decentralized, self-organized way, using network coordinates to facilitate inter-peer distance estimation. We discuss the benefits of super-peer infrastructures in the context of Desktop Grids and present the outcome of experiments conducted on PlanetLab to observe the constructed topology's behavior in a practical environment.

1 Introduction

Desktop Grids have been proposed to leverage the idle computation time provided by desktop computers [1, 11, 12, 18]. The construction of Grid structures based on Peer-to-Peer (P2P) overlay networks generally fosters scalability in large-scale settings [9]. The scalability gain arises from the absence of a central authority, leading to the issue of effective resource discovery and utilization. Nodes will need to communicate with their fellow peers to keep track of available resources, handle job submissions and care for their completion. This joint effort may be directed by a middleware for distributed computing. When run on every participating machine, it enables peers to coordinate their actions, allowing applications to have arbitrary jobs processed by the Desktop Grid in a distributed fashion. In this paper, we consider a way to dynamically build and maintain self-organized P2P overlay networks with integrated super-peer infrastructures for improved scalability as a key building block of such middleware.

Well-known properties of fully decentralized P2P systems include self-organizing and fault-tolerant behavior. However, the scalability of such networks may become an issue in the case of substantial growth when inefficient communication schemes prevent the network from growing smoothly [10, 16]. A feasible solution to this issue is the introduction of *super-peers* [25]. Super-peers are peers that act as relays 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 substantially smaller number of links than before [10, 14, 25].

Generally, we strive for a topology in which a subset of the nodes will function as super-peers while the remaining nodes, henceforth called edge peers, are each assigned to exactly one of the super-peers, respectively. Super-peers 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 overlay communication via their particular super-peer. As a side effect of this approach, a super-peer's proxy function helps to circumvent its edge peers' firewalls that may prevent direct communication between edge peers. The set of super-peers C makes up the *core*.

The distributed construction of a super-peer topology may require delay measurements between nodes. Sending delay probe packets (*ping* packets) between all nodes would spark a load surge in the network, severely hampering scalability, particularly in large-scale environments. Network coordinates effectively tackle this issue. These are space coordinates individually assigned to every node such that the spatial distance between any two nodes approximates the true network delay between those nodes. We resort to network coordinates for a scalable, low-load approach to delay estimation.

The remainder of this paper is organized as follows. In Section 2, we briefly outline our middleware for distributed computing. In Section 3, we provide a protocol for super-

978-0-7695-3156-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CCGRID.2008.51

peer topology construction and maintenance, including a distributed algorithm that promotes suitable peers to super-peers. In Section 4, we discuss the utility of network coordinates in this context. In Section 5, we state the scenario, setup and results of our experiments on PlanetLab. Section 6 contains an overview of related work. The paper concludes with directions for future research in Section 7.

2 Middleware concept

We are in the process of developing a middleware for P2P-based distributed computation. Built on the Java platform for portability reasons, this middleware is intended to run on every node willing to join arbitrary distributed computation projects. Every participating node is supposed to be ready to provide computational resources to the common cause, and may also act as a provider of jobs when available. In this context, nodes become peers in an overlay network. Peers communicate by asynchronously exchanging messages.

The middleware shall enable the formation of a Desktop Grid by offering job processing services to applications. Essentially, the middleware will accept job information passed by an application, identify a sufficiently large set of peers willing to act as job contractors (workers), temporarily set up a secondary overlay to handle the communication with these peers, and support the application in assembling the returned results after the distributed computation effort has terminated. If required, workers may communicate with each other to exchange intermediate solutions.

In this paper, we focus on the super-peer topology construction and maintenance mechanism, a particular component of the middleware which cares for the setup and operation of the underlying P2P overlay network.

3 Super-peer topology

3.1 Maintenance protocol

We have designed and implemented a protocol for super-peer overlay network construction and maintenance [15]. The middleware uses this protocol to build a P2P structure underneath itself.

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 QUERYSUPERPEER message to Y. Y responds with a list of all super-peers it knows, wrapped in a SUPERPEERLIST 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 CONNECT message to Z. (Y and Z may be identical.) Depending on its state, Z replies with either ACCEPT or REJECT. In the case of ACCEPT, X is now linked to Z as one of its edge peers, and as such, X has been accepted to join the overlay, while in the case of REJECT, X will pick another super-peer from the list and try again. When X wants to disconnect from Z later, X sends a DISCONNECT 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 NEWSUPERPEER message to L. If L accepts the offer, it replies with an ACCEPTSUPERPEER message. Eventually, K broadcasts L's presence by transmitting INSERTSUPERPEER messages to its fellow 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 arises if K determines one of its edge peers to be a better super-peer than itself. In both cases, K transmits a REJECT message to each of its edge peers, disconnecting them. K also sends REMOVESUPERPEER messages to all super-peers which in turn may forward this information to their respective edge peers. In their next reorganization phase, freshly disconnected edge peers will connect to other super-peers, rejoining the overlay. An edge peer changes its super-peer by sending a CONNECT message to the new super-peer and, in case it is accepted, a DISCONNECT message to the old super-peer.

Super-peers keep track of core memberships and provide edge peers with lists of super-peers, particularly when a prospective edge peer is rejected.

3.2 Super-Peer Selection Algorithm

We have developed a distributed Super-Peer Selection Algorithm (SPSA) as an integral part of the aforementioned protocol, aiming to minimize the total communication cost [15]. In a P2P setting, this cost can be thought of as the total all-pairs end-to-end communication delay. To ensure overlay scalability, SPSA aims for $\sqrt{n}$ super-peers and $\sqrt{n}-1$ edge peers per super-peer, with n being the total number of peers in the overlay which may be estimated by any peer using the size of its most recent super-peer list as an approximation of $\sqrt{n}$. With $\sqrt{n}$ super-peers, the maximum number of connections a super-peer needs to keep up is minimized.

SPSA is meant to run on every peer in the overlay. With SPSA, a peer regularly enters a reorganization cycle every γ seconds. In the reorganization cycle, an edge peer changes its super-peer if it finds another super-peer to be closer. Fig. 1 contains SPSA's edge peer part. A super-peer either stays put or performs a reorganization step. A super-peer K's reorganization step consists of a choice among three options, based on its current estimate x of total peers in the overlay and the set E_K of its assigned peers including itself. K downgrades to edge peer level if it finds the number of its assigned peers including himself, $|E_K|$, too low

($|E_K| < 1/2 \cdot \sqrt{x}$). Conversely, if $|E_K|$ exceeds the maximum load threshold ($|E_K| > 2 \cdot \sqrt{x}$), K picks one of its edge peers, J, and promotes it to super-peer level. In this case, J may be any edge peer whose promotion will split K's edge peer set such that neither K nor J will decide to downgrade due to insufficient load. A third reorganization option besides downgrade and promotion is replacement. With this option, K downgrades, becoming an edge peer while promoting one of its edge peers, J, to become super-peer, and relocates its currently supported set of peers to J if it finds J to be able to deliver the super-peer service at lower cost. This aspect is quantified by the notion of gain, defined as the difference of K's cost of supporting the current edge peer set and the cost that J would bear. K picks J as the edge peer that maximizes the gain. If the gain is negative for all of K's edge peers, K will not replace itself during this cycle. The super-peer reorganization phase is depicted in Fig. 2 for a super-peer K, with C as the set of known super-peers and E_K as the given set of peers assigned to K (including K itself).

SPSA operates in a distributed way. No global view is required, and the algorithm seamlessly fits into the super-peer topology maintenance protocol.

4 Network coordinates

Given a metric space, nodes can be embedded into that space such that the spatial distance between any two nodes approximates the measurable delay between those nodes in the network. The coordinates that every node individually attains by this embedding are called network coordinates. Once a node has learned of another node's network 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). Delay estimation using network coordinates causes only a fraction of the network load which would be incurred with standard delay measurement methods, even in the face of additional administrative overhead [4, 13, 17, 20].

Network coordinates mechanisms embed a network into a metric space which is often Euclidean. In contrast to such spaces, live networks frequently violate the triangle inequality due to routing policies. For this reason, error-free embeddings are practically unavailable [26]. Furthermore, algorithms used to embed the network introduce an additional embedding error if they fail to locate an optimum solution.

```
every γ seconds do:
  I ← nearestSuperPeer()
  if mysuperpeer ≠ I then
    mysuperpeer ← I
    connectTo(I)
  end if
```

Figure 1. SPSA, edge peer part

```
every γ seconds do:
  if |E_K| < 1/2 · |C| then {downgrade: become an edge peer}
    role_K ← edge-peer
  else if |E_K| > 2 · |C| then {promotion: choose a new super-peer}
    J ← most suitable edge peer
    role_J ← super-peer
  else {replacement: try another super-peer}
    J ← most suitable edge peer
    if replacement with J yields gain then
      role_K ← edge peer
      role_J ← super-peer
    end if
  end if
```

Figure 2. SPSA, super-peer part

For network coordinates to work, nodes require both their own coordinates and the coordinates of any remote node for which the distance needs to be estimated. Node coordinates may be disseminated via background gossip and/or piggybacked on existing messages. In this setting, merely the super-peers' coordinates need to be spread to all nodes. The edge peers' coordinates are only required by their super-peer.

We have resorted to the Vivaldi system, a network coordinates generation mechanism which relates to the effect of spring deformation [5]. It can be initialized by providing arbitrary coordinates to every participating node, or alternatively, having all nodes start from the origin. With Vivaldi, two communicating nodes measure both their link's delay and their spatial distance, then set up a virtual spring between them. 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. Vivaldi will strive to reduce that energy for all springs to a minimum, preferably zero, level.

5 Experiments

5.1 PlanetLab

We found PlanetLab [3, 22] to be a valuable intermediate stepping stone between simulation and unrestricted,

practical deployment of distributed applications in the open Internet. While PlanetLab's nodes are linked via the Internet, exposing messages to detrimental phenomena like delay and loss, the system offers auxiliary facilities to ease experiments, including a limited global view with control over all participating peers and high-throughput inter-site links [2]. On the way to a full-scale real deployment, we have chosen PlanetLab as an established environment to conduct experiments. The following subsection shall provide an in-depth description of the experiments' configuration.

5.2 Configuration

Preliminary simulations of SPSA's behavior have suggested promising performance [15], however these simulations were conducted under a number of assumptions that cannot be upheld in a practical environment. These assumptions included the absence of message loss and reordering, sudden node departure, queueing and computing delays, and static round-trip times that did not change over time. PlanetLab has provided us with a facility to sample and quantify the effectiveness of overlay topologies created by the middleware-backed peers in a live setting.

For benchmark purposes, we have computed the value of an objective function Z that returns the total communication cost in symmetric networks, assembled from the total cost of edge peer-super-peer links and intra-core links according to a given super-peer configuration. Formally, Z may be written as

$$Z = \overbrace{2 \cdot (n-1) \cdot \sum_{i,k} d_{i,k} \cdot x_{i,k}}^{\text{edge peer connections}} + \underbrace{\sum_{\substack{k,m \\ k<m}} d_{k,m} \cdot x_{k,k} \cdot x_{m,m} \cdot 2 \cdot |E_k| \cdot |E_m|}_{\text{intra-core connections}}$$

where $d_{i,k}$ gives the distance (the true round-trip time as measured on PlanetLab) between nodes i and k, while $x_{i,k}$ represents a binary variable which is set to 1 if node i, as an edge peer, is connected to super-peer k. For any super-peer k, $x_{k,k} = 1$ holds. The weights of edge peer-super-peer links are multiplied with $(n-1)$ to reflect the total all-pairs cost view. Intra-core links are weighted by a factor of $|E_k| \cdot |E_m|$ where $|E_x|$ denotes the number of edge peers assigned to the super-peer x plus the super-peer itself. In both cases, links are used bidirectionally, accounting for the factor 2, respectively.

On PlanetLab, we have monitored in particular the number of super-peers and the value of the objective function, Z, over time. A limited global view has been established by preselecting the participating peers. For our experiments, we have resorted to CoMon [19] to narrow the peer set to a number of systems which have been reported to be available, operational, and not exposed to heavy load. Additionally, participating nodes were required to be responsive to incoming delay measurements, ultimately resulting in a set of 52 PlanetLab nodes to make up the experiments' peer set. These peers regularly reported their state to a supervising machine outside PlanetLab. Peers continuously measured round-trip times to other peers as input to Z.

Each experiment run lasted for 10 minutes, with a total number of 20 runs. Reorganization cycles were scheduled individually for each node to occur every $\gamma = 4$ seconds. In the beginning, one random peer was chosen to act as the first super-peer. Subsequent super-peers would be selected according to the topology maintenance protocol.

With respect to Vivaldi, we have initially placed the nodes at positions drawn from a uniform random distribution. Each coordinate was contained in the range [0, 500], such that in the 4-dimensional space that has been used, the initial gap between any two nodes did not exceed 1000 units of distance. 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 2 seconds, routed to a random destination via the super-peers, which then established the pairwise force using a three-way handshake. Accordingly, the spatial distance between two nodes served as an approximation to the true delay, measured in milliseconds. The list of super-peers carried by every peer contained both super-peer network addresses and network coordinates.

In line with the SPSA target, we have expected the number of super-peers to approach $\sqrt{n}$ given $n = 52$ under optimal circumstances, resulting in 7 to 8 super-peers for a 52-peer overlay network.

5.3 Results

Table 1 contains a breakdown of the experiments' aggregated outcome with respect to the objective function (mean and standard deviation) and number of super-peers (median, minimum and maximum amount).

After creating additional super-peers in the beginning, the decentralized process quickly converged, approaching a reasonably stable state after approximately 300 seconds of simulated time. Judging from these results, SPSA has proved to work in practice, reaching the expected number of 7 super-peers, as depicted in Table 1.

The decreasing course of the objective function, Z, is shown in Fig. 3. Oscillations in the objective function stemmed primarily from fluctuating network coordinates and changes in the super-peer selection.

Time	Objective function Mean	Std Dev	Number of super-peers Median	Min	Max
0	–	–	1	1	1
60	678078	521128	7	6	9
120	649697	303142	7	6	8
180	538445	97051	7	6	10
240	505364	83060	7	5	8
300	477589	68401	7,5	6	9
360	462334	49179	7	6	9
420	475461	80944	7	6	12
480	468039	80124	7	6	13
540	483291	81462	7	6	11
600	469006	68002	7	6	13

Table 1. Results

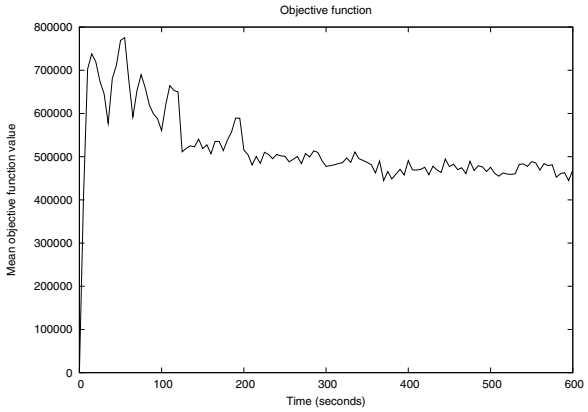

Figure 3. Mean objective function value

6 Related work

Initial approaches to the matter of distributed computation on desktop systems have tended to include central entities for coordination purposes. Among these approaches, the now discontinued POPCORN system [18] provided a Java-based architecture for CPU time markets. The CORBA-based InteGrade middleware uses clusters of machines coordinated by hierarchically arranged manager nodes [7] but does not incorporate a dynamic topology-aware structure.

Recent developments in the field of P2P-based Desktop Grids include P3 [23] and Vishwa [21]. P3 resorts to JXTA [8] for topology construction and maintenance which, as of now, does not perform latency-related overlay optimization, while Vishwa nodes form clusters using hop count as the primary distance metric, and establish links to close peers according to the network delay. However, there is no dynamic, self-organizing hierarchical peer infrastructure in Vishwa as there is in our approach.

The relationship between Grid and P2P systems has attracted considerable attention. In [24], the authors find Grids to essentially be P2P systems. Likewise, the research presented in [6] expects Grid and P2P structures to eventually converge mainly because of their common goal, resource sharing within virtual organizations. Our middleware will provide Grid-like facilities based on a P2P structure, merging the two concepts.

With respect to super-peer topology construction, an approach closely related to ours is [10] which also proposes building a super-peer overlay topology using network coordinates. The suggested mechanism, SG-2, mimics the social behavior of insects to promote particular nodes to super-peer level and vice versa when appropriate. Nodes observe capacity constraints and a maximum tolerable latency regarding connections between edge peers and super-peers. Like our approach, SG-2 builds a super-peer topology in a decentralized way by local interaction. It strives to minimize the number of super-peers under given constraints, however according to [10], this number is on the order of $\mathcal{O}(n)$ while our mechanism achieves $\mathcal{O}(\sqrt{n})$, with n being the total number of peers. Moreover, because of the spatially limited zones of influence, SG-2 does not ensure overlay connectivity. Technically, both approaches resort to Vivaldi as a provider for network coordinates.

7 Conclusion

Desktop Grids operating on super-peer overlay structures benefit from improved scalability by having the super-peers handle administrative tasks. Furthermore, the absence of a central server enhances overall robustness. This paper has focused on PlanetLab experiments with a vital part of a new middleware for distributed computing currently under development. Built for scalability, it constructs and operates a super-peer overlay topology in a self-organized, distributed way. In this context, we have concentrated on the middleware's topology construction and maintenance mechanism with emphasis on the benefits of super-peer structures for Desktop Grids. The experiments have proved that SPSA indeed creates the desired number of super-peers, setting up an efficient overlay topology in a live setting.

Future work will focus on adding job-handling capabilities to the middleware. Moreover, we plan to consider economically rational peers instead of assuming all peers to be altrustic, and examine a DHT-type structure to connect the super-peers in order to reduce the number of intra-core links.

References

[1] D. P. Anderson and G. Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)*, pages 73–80, 2006.

[2] 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*, 2004.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[4] 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.

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

[6] I. T. Foster and A. Iamnitchi. On death, taxes, and the convergence of Peer-to-Peer and Grid Computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 118–128, 2003.

[7] A. Goldchleger, F. Kon, A. Goldman, M. Finger, and G. C. Bezerra. InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. *Concurrency and Computation: Practice and Experience*, 16(5):449–459, March 2004.

[8] L. Gong. JXTA: A network programming environment. *IEEE Internet Computing*, 5(3):88–95, 2001.

[9] A. Iamnitchi and I. T. Foster. A Peer-to-Peer Approach to Resource Location in Grid Environments. In J. Nabrzynski, J. M. Schopf, and J. Weglarz, editors, *Grid resource management: state of the art and future trends*, pages 413–429. Kluwer, 2004.

[10] 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.

[11] D. Kondo, G. Fedak, F. Cappello, A. A. Chien, and H. Casanova. Characterizing Resource Availability in Enterprise Desktop Grids. *Future Generation Computer Systems*, 23(7):888–903, Jan. 2007.

[12] D. Kondo, M. Taufer, C. L. Brooks, H. Casanova, and A. A. Chien. Characterizing and Evaluating Desktop Grids: An Empirical Study. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, 2004.

[13] 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.

[14] 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, Lecture Notes in Computer Science*, volume 3252, pages 221–228, 2004.

[15] P. Merz, M. Priebe, and S. Wolf. A simulation framework for distributed super-peer topology construction using network coordinates. In *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 491–498, 2008.

[16] 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.

[17] T. S. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM*, pages 170–179, June 2002.

[18] N. Nisan, S. London, O. Regev, and N. Camiel. Globally Distributed Computation over the Internet - The POPCORN Project. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS 1998)*, pages 592–601, 1998.

[19] K. Park and V. S. Pai. CoMon: a mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Operating Systems Review*, 40(1):65–74, Jan. 2006.

[20] P. R. Pietzuch, J. Ledlie, M. Mitzenmacher, and M. I. Seltzer. Network-Aware Overlays with Network Coordinates. In *Proceedings of the 1st Workshop on Dynamic Distributed Systems*, July 2006.

[21] M. V. Reddy, A. V. Srinivas, T. Gopinath, and D. Janakiram. Vishwa: A reconfigurable P2P middleware for grid computations. In *Proceedings of the 2006 International Conference on Parallel Processing (ICPP)*, pages 381–390, 2006.

[22] T. Roscoe. The PlanetLab Platform. In R. Steinmetz and K. Wehrle, editors, *Peer-to-Peer Systems and Applications*, pages 567–581. Springer, 2005.

[23] K. Shudo, Y. Tanaka, and S. Sekiguchi. P3: P2P-based middleware enabling transfer and aggregation of computational resources. In *Proceedings of the 5th International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pages 259–266, 2005.

[24] D. Talia and P. Trunfio. Toward a Synergy Between P2P and Grids. *IEEE Internet Computing*, 7(4):94–96, July 2003.

[25] 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.

[26] 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.