A P2P-based Framework for Distributed Network Management

Andreas Binzenhöfer[1], Kurt Tutschku[1], Björn auf dem Graben[1], Markus Fiedler[2], and Patrik Arlos[2]

[1] Department of Distributed Systems
Institute of Computer Science
University of Würzburg, Am Hubland, 97074 Würzburg, Germany
{binzenhoefer, tutschku, adgraben}@informatik.uni-wuerzburg.de
[2] Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
371 79 Karlskrona, Sweden
{markus.fiedler, patrik.arlos}@bth.se

Abstract. In this paper we present a novel framework supporting distributed network management using a self-organizing peer-to-peer overlay network. The overlay consists of several *Distributed Network Agents* which can perform distributed tests and distributed monitoring for fault and performance management. In that way, the concept is able to overcome disadvantages that come along with a central management unit, like lack of scalability and reliability.
So far, little attention has been payed to the quality of service experienced by the end user. Our self-organizing management overlay provides a reliable and scalable basis for distributed tests that incorporate the end user. The use of a distributed, self-organizing software will also reduce capital and operational expenditures of the operator since fewer entities have to be installed and operated.

1 Introduction

A peer-to-peer (P2P) system is a highly distributed application architecture. The underlying technology has so far only received a doubtful reputation due to its use in file sharing applications. P2P algorithms, however, might be highly helpful in implementing novel distributed, self-structuring network management concepts. In this work we suggest the application of a current generation, structured P2P overlay network for fault and performance management with the aim of enhancing conventional management functions.

In general the goal of Network Management is "to ensure that the users of a network receive the information technology services with the quality that they expect" [1]. However, monitoring and provisioning of that quality in an end-to-end manner as perceived by a user [2] is rarely achieved. The monitoring is usually carried out by rather centralized entities (Network Management Systems) and only in those parts of the network a provider is responsible for. Coordination of the monitoring among different administrative domains is rarely achieved, which also affects possibilities to locate faults and to evaluate end-to-end QoS.

A central fault testing and QoS monitoring architecture typically results in additional, complex entities at the provider. The operator has to ensure the reliability of the entities and assess their scalability. The systems have to scale with $O(N^2)$ due to the $N(N-1)$ potential relationships among N end systems. In addition, relaying monitoring data consumes bandwidth, delays its availability, and might get lost in case of a network failure. A decentralized QoS monitoring, as for example, located on the user's end system, might avoid these disadvantages. The use of a distributed, self-organizing software will reduce capital and operational expenditures (CAPEX and OPEX) of the operator since fewer entities have to be installed and operated. Scalability can be achieved by re-using resident resources in conjunction with local decisions and transmission of less data.

We propose a new, distributed, self-organizing, generic testing and QoS monitoring architecture for IP networks. The architecture will complement today's solutions for central configuration and fault management such as HP OpenView [3] and IBM Tivoli [4]. The architecture is based on equal agents, denoted as Distributed Network Agents (DNA), which form a management overlay for the service. In this context the word *agent* is not to be understood as an agent as used by the Artificial Intelligence community, but rather as a piece of software running on different peers, like, e.g., an SNMP-Agent. The self-organization of the overlay is achieved by Kademlia [5], a P2P-based Distributed Hash Table (DHT).

The suggested architecture facilitates the autonomic communication concept [6] by locally determining the perceived QoS of the user from distributed measurements and by exploiting the self-organization capabilities of the DHT for structuring the overlay. It will be able to communicate with standard-NMS via well-established interfaces. Thus, it can be seen as a QoS-enabling complement of existing Network Management solutions.

The remainder of the paper is structured as follows: Section 2 introduces the architecture of a DNA and shows how the framework can be used for local and distributed tests. In Section 3 we give an overview of the current P2P generation and motivate why we chose Kademlia as the basis of the DNA overlay. Some details about the implementation of our prototype will be given in Section 4. The functionality of the DNA is validated by simulation in Section 5. Section 6 finally concludes the paper and summarizes our future work.

2 The DNA Framework

The DNA framework represents an independent distributed application intended to support the central network monitoring station. In general a central monitoring entity has three major disadvantages:

– It is a single point of failure. Once the single central monitoring unit fails, the network will lose its control entity and will be without surveillance. The same problem could, e.g., be caused by a distributed denial of service attack. That is, the functionality of the entire network management depends on the functionality of a single central unit.
– It does not scale. On the one hand the number of hosts that can be monitored at a given time is limited by the bandwidth and the processing power of the central monitoring unit. On the other hand there is a growing number of services that has to be monitored on each host due to the diversity of services that emerge during the evolution of the Internet.
– It has a limited view of the network. While a central network manager is able to monitor, e.g., client A and Server B, it has hardly any means of knowing the current status of the connection between the two monitored devices themselves.

The DNA application is able to support the central unit in two ways. At first, the DNAs constantly monitor the network in a distributed way and send a message back to the central server in case the condition for some trigger event is met, similar to SNMP traps. Secondly the central server can query the current state of the DNA on demand. In case the central server fails, the DNAs will still be functional and can store the gathered information until the central server goes back online again. First ideas have been discussed briefly at [7]. The DNA framework can be used as a plug and play platform for overlay network monitoring approaches like [8] and [9].

2.1 The Basic Concept

The Distributed Network Agent (DNA) is based on a modular concept shown in Figure 1. The main component of the DNA, the so-called Mediator, runs as a daemon in the background and is responsible for the communication between the user and the individual test modules. A test module consist of several tests that are similar in respect of their functionality. They represent the functionality of the DNA and can be added or removed without any influence on the operability of the DNA architecture. The user is able to connect to the Mediator using the graphical user interface (GUI) or the command line. He can manually start tests or read the results of tests that have already been performed. Figure 1 summarizes the design of the DNA framework. The Mediator either receives a test request from the user or autonomically schedules a test itself. It then performs the corresponding test using the provided Interfaces and finally sends the results to the GUI upon request. Any test module that implements all features required by the Interface component can be added to the DNA framework. A description of the features required by the Interface component and their purpose is given in the following:

– *Default Tests:* Each test module has to provide a list including all tests that will be run when the module is called by the Mediator without any

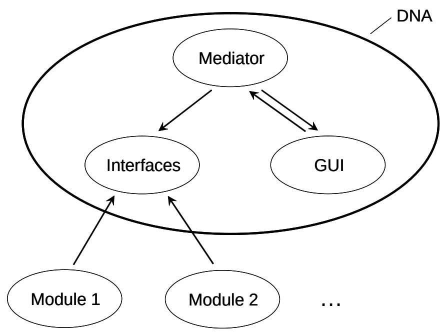

Fig. 1. The modular framework of the DNA

parameters. This list of default tests will be executed when the user did not specify any details. Advanced users, however, are able to choose another subset of tests from the module that is adapted to their specific needs. The default test sequence offers the possibility to implement a test module that, e.g., includes tests dealing with different kinds of IP configuration. In this example the default test sequence could cover all properties of a static IP, while support for DHCP might be optional.

- *Test Dependencies:* If not specified otherwise all selected tests of a test module are performed in parallel. However, the DNA offers the possibility to use test dependencies. That is, a test module can provide a list containing the dependencies of its individual tests. The Mediator does not start a test until all its dependencies were performed successfully. The following two tests provide a simple example for test dependencies: A test that checks the state of a network interface card (NIC) and a second test that pings a predefined host using the same NIC. Obviously the second test is dependent on the first one, since it can only succeed if the NIC is up and running. The Mediator would therefore only execute the second test if the first test already finished successfully.

Since a great fraction of network problems is actually caused by local errors like misconfiguration and software failures, we outline a test module containing local tests in the following section. Before a DNA takes part in the DNA overlay, it can run the local test module to eliminate any possibility of local errors.

2.2 Local Tests

In this section we briefly describe a test module containing local tests, which is an inherent part of our prototype discussed in Section 4. First, we summarize the set of default tests and give a brief description of the remaining tests in the module afterwards. All local tests are bound to a specific network interface card, which can be selected by the user.

Default Test Sequence The default test sequence contains eight tests, which can be divided into "Hardware and TCP/IP" and "Local configuration and network connectivity". The five tests in the latter category all depend on the three tests dealing with hardware and TCP/IP.

Hardware and TCP/IP

– *NICStatus:* This test returns information about the network interface card and its current state, including a driver check and the like. It is mainly used to eliminate the possibility of hardware failures or problems with the driver.
– *NetConnectionStatus:* This test is used to check the current connection status of the NIC. Causes for an error include a cable that is not plugged in, or a network interface card still running an authentication process.
– *PingLocalHost:* The functionality of the TCP/IP stack is validated by sending a ping request to the loopback address (127.0.0.1).

In case all three tests finish successfully, the Mediator will call the remaining five tests of the default test sequence.

Local configuration and network connectivity

– *IPConfiguration:* This test verifies that a valid IP address is assigned to the NIC. It also checks if the associated gateway is on the same subnet as the IP address.
– *DNSConfiguration:* The test verifies that at least one DNS server is assigned and can be reached by a ping request. Furthermore, the functionality of the DNS server is tested performing a predefined DNS lookup and optionally a reverse lookup. If the ping message failed whereas the lookup was successful, no warning message is reported.
– *DHCPLease:* In case of using DHCP on a machine running Windows, the IP address is checked to ensure that the network interface card is not set up to use a so called Automatic Private IP Address (APIPA). This might occur, if the server has no more capacities to provide a new IP address or if the user participates in an encrypted wireless LAN (e.g. WEP) using an invalid key.
– *PingOwnIP:* To exclude a communication problem between the operating system and the NIC, the IP address assigned to the adapter is pinged.
– *PingWellKnownHosts:* This tests sends a ping request to a list of predefined well known hosts. If a specified number of hosts in this list does not respond in time, the test returns an error.

Additional Local Tests The following three tests belong to our local test module but are not part of the default test sequence, since they are specific to the Windows OS. Advanced users can include the tests in case the DNA is running on a Windows based platform.

– *EventViewer:* The test searches in the "Event Viewer Log" for error events caused by TCP/IP or DHCP. If the Windows event viewer has recorded problems with respect to TCP/IP or DHCP those errors will be forwarded to the GUI of the DNA.

– *HostsAndLmHosts:* Before Windows uses DNS or WINS it tries to resolve a domain name using the HOSTS or LMHOSTS file. If one of these files contains a wrong entry the resolution of the corresponding name fails. The test searches for syntax errors in both files and sends ping requests to valid entries.
– *RoutingTable:* The Windows routing table is divided into a dynamic and a persistent part. This test pings the gateways of both tables and reports an error message, if one of them is not reachable.

2.3 Distributed Tests

In this section we describe how to use the DNA framework to implement distributed test modules. A distributed test is a test that is performed in conjunction with at least one other DNA. To be able to communicate with each other the DNAs build an overlay network on top of the monitored network. To perform a distributed test a DNA can then either connect to a randomly chosen DNA or to a specific DNA chosen by the user. Section 3 describes the P2P based DNA overlay in detail.

The following two distributed tests point out the possibilities of the distributed DNA framework that arise by extending a simple local test to a distributed test:

– *PingWellKnownHosts:* If a single DNA or a central network manager does not receive a ping reply from a well known host, either the host or any link on the path to this host could be down. Using the DNA framework, however, a DNA can ask another DNA to ping the same host and evaluate the returned result. In case another DNA is able to ping this host, the possibility that this host is down can be ruled out and the cause of the error can be narrowed down to a network problem between the DNA and the well known host. If the DNA has knowledge about the network topology, which could, e.g., be gained using network tomography, the distributed ping test can also be used to pinpoint the broken link or to locate a bottleneck by comparing the delay of the ping messages.
– *DNSProxy:* In general a DNA can use another DNA as a temporary proxy or relay host. In case a DNA loses the connection to its DNS server and can thus no longer resolve domain names, it can use another DNA as a DNS proxy. That is, the DNA forwards the DNS query to another DNA that in turn tries to resolve the domain name using its own DNS server. This way the DNA is able to bridge the time its DNS server does not respond to DNS queries. In a similar way two DNAs with a broken direct connection could use a third DNA that still has a connection to both DNAs, as a temporary relay host.

As stated above the DNAs build an overlay network to be able to communicate with each other. They are able to communicate with a random DNA in the overlay or to search for a specific DNA. The following two tests provide examples of how to build distributed applications based on this aspects of the DNA framework:

- *PortScan:* If a DNA peer is running a webserver or offers some other service that requires an open port it usually is probed by a central network manager to ensure a continuous service. On the one hand this method does not scale with the number of services the central network manager has to monitor, on the other hand the peer running the service has no influence on the time of the next check. Using the DNA framework, however, the peer is able to ask a random DNA to see if it can reach the offered services. This is also a scalable way to monitor a large number of services. The DNAs monitor the services running on their peers in a distributed way and only send a message back to the central network manager in case of an error.
- *Throughput:* Usually it is not easy for a user to verify a service level agreement or to measure the bandwidth to another point in the network. The possibility to search for a specific DNA enables a peer taking part in the DNA overlay network to search for a specific communication partner and ask for a throughput test. A very simple way to do so is to constantly send traffic to the other DNA for a certain period of time and simply measure the average throughput. However, there are more sophisticated ways, which we intend to integrate in future work.

The above tests are just some examples of how to use the DNA framework. A future possibility consists in passive monitoring of data streams between the peers at both peers and exchanging the measurement results in order to determine potential bottlenecks as described in [10]. In Section 6 we summarize our ideas for a new, distributed passive QoS monitoring concept. The next section discusses general security issues and a way of how to deploy new test modules to the DNA overlay network.

2.4 Deployment of New Tests

Considering a running DNA overlay network, one can not assume that all DNAs are always having the same test modules. An obvious way to deploy new test modules is to send the modules on demand. That is, if A asks B to perform a distributed test but B does not have this specific test, A simply sends the test module to B. However, this implies that B implicitly trusts A. This is a security risk that is obviously not negligible. In fact a framework that allows other machines to run arbitrary code would be the perfect tool for distributed denial of service attacks.

One way to solve this problem is to only download new test modules from a central trust server. That is, all DNAs trust a central entity and only run code that is signed by this central authority. While this solution is sufficient for small networks it does not scale to larger networks. A scalable implementation of the DNA framework therefore needs a distributed trust model. Since there is an independent research area dealing with security and this paper is mainly intended to be a proof-of-concept for a P2P based framework for network monitoring, we will refrain from addressing security issues. There exist, however, different approaches to build distributed trust models for P2P systems. In [11], e.g., Josephson proposes a scalable method for P2P authentication with a distributed single sign-on service, that could be used as the trust model for our DNA.

3 The P2P-based DNA Overlay

To be able to search for other peers, the individual DNAs build an overlay network. The main purpose of this overlay is to keep the DNAs connected in one logical network and to enable a single DNA to find another DNA in reasonable time. The current generation of structured P2P networks is able to locate other peers using only $O(\log(n))$ messages while keeping connections to only $O(\log(n))$ other peers in an overlay network of size n [12]. We chose the Kademlia algorithm [5] as the basis of the DNA overlay. Kademlia offers a set of features that are certainly not unique to it, but that are so far not offered by another single algorithm all at once. In detail those features are:

1. *Symmetry:* Due to the symmetry of the XOR metric $d(x, y) = d(y, x)$, the DNA overlay network is symmetric as well.
2. *Unidirectionality:* For any identifier x and an arbitrary distance $s > 0$ there is exactly one point y such that $d(x, y) = s$. Thus, independent of the originating peer, lookups for the same peer will all converge along the same path.
3. *Parallel queries:* One of the most advantageous features is the possibility to send out parallel queries for the same key to different peers. This way, timeouts on one path do not necessarily delay the search process, guaranteeing faster and more reliable searches under high churn[3] rates.
4. *Arbitrary Neighbors*: In [5] neighbors in the overlay were chosen by the time of last contact to obtain more reliable connections. Neighbors, however, can be chosen by any criterion like trustability or reliability. The best known approach is to chose peers according to their ping times to guarantee low latency paths when searching.
5. *Low periodic traffic:* In contrast to most other algorithms Kademlia uses almost no periodic overhead traffic but exploits the search traffic to stabilize the overlay network connections. Configuration information spreads automatically as a side-effect of key lookups.
6. *Security:* As a result of its decentralized nature Kademlia is resistant against denial of service attacks. The security against attackers can even be improved by banning misbehaving peers from the peers buckets.

The DNA overlay enables fast searches for random and specific communication partners. In Section 5 we discuss results obtained from our simulator in detail.

[3] The rate at which peers join and leave the overlay network

4 The DNA Prototype

As a proof of concept and practicability of our work, we implemented a prototype of the DNA. While the general concept is platform independent, the implementation was done in *.NET*, as the WMI Interface offers the opportunity to access all kind of information about the local system state as well as the state of the network in very few lines of code. The DNA prototype was implemented within the scope of an industrial cooperation with Datev, one of the largest information service providers and software firms in Germany as well as Europe. This provided us with the opportunity to prove the functionality of the concept by successfully running the DNA in a realistic testbed with over 50 machines. One of the main advantages besides those mentioned in the previous sections turned out to be the plug and play character of the DNA framework. Due to the Kademlia based overlay network, the DNA framework is self-configuring. To include a new DNA into the existing overlay, the user just has to start the client and it will automatically find its position in the overlay network. On the other hand, if a client fails, the overlay network proved to be self-healing by automatically updating the neighbor-pointers needed to keep the overlay network stable. The next section contains a simulative evaluation of the stability of the proposed solution.

5 Simulation Results

In this section we prove the functionality and the scalability of our DNA prototype by simulation. The simulator is written in *.NET*, based on the code used for the prototype. The network transmission time for one hop was chosen according to an exponential distribution with a mean of 50 ms. If not stated otherwise, we let a number of nodes join the overlay network and begin a churn phase once the overlay has initially stabilized. To generate churn we model the online time of a peer by means of an exponentially distributed random variable. The longer a peer stays online on average, the less churn there is in the overlay network. To provide credible simulation results, we produced several simulation runs and calculated the mean as well as the corresponding confidence intervals.

To show the scalability of the DNA we regard the time needed to complete a search for other peers in dependence of the overlay size. First, we regard a system without any churn, i.e. we let n DNAs join the system, wait until the overlay network stabilizes and then let a number of random DNAs search for other peers. The concave curves in Figure 2 show that the search in our prototype does indeed scale. In a network of 1000 peers a search for another DNA takes less than 200 ms. To see the influence of churn on the search time, we repeated the same simulations and set the average online time of a peer to 60 minutes. To keep the size of the overlay constant on average, we chose a corresponding Poisson arrival process to let new DNAs join the network. The results are also shown in Figure 2. Due to timeouts caused by the churn in the system, searches take longer than in the scenario without churn. However, as seen from the concave

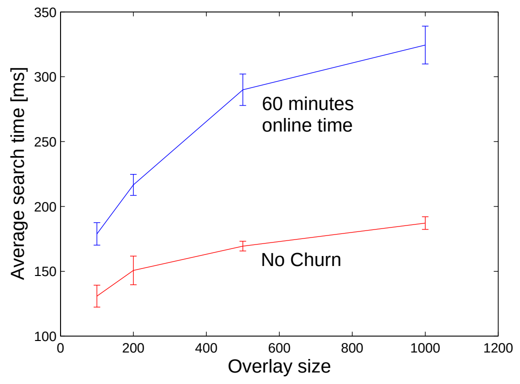

Fig. 2. Duration of a search as a function of the overlay size

curves, the search algorithm scales when the system becomes larger and enables fast searches for other peers.

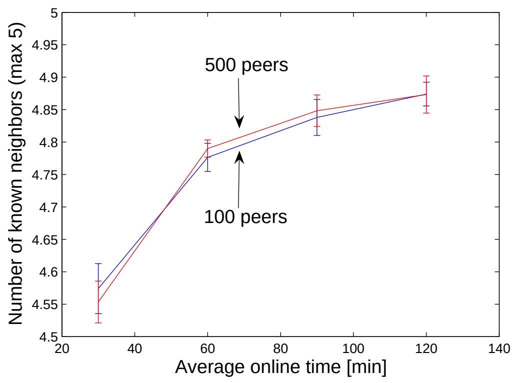

Fig. 3. Average number of known direct neighbors

In previous studies [13, 14] we showed that the size of the network itself is not the crucial factor in terms of scalability and overlay stability. In fact the robustness of the overlay is mainly influenced by the current churn rate. A good way to prove the stability of the overlay network is therefore to look at the correctness of the neighbors of a peer under churn. In general the functionality of a structured P2P-overlay can be guaranteed, as long as the information about a peers neighborhood is not lost. In case the information about more distant peers is lost, the performance of the overlay might get slightly worse, but the underlying algorithms will still be functional. We study the correctness of the direct neighbors to evaluate the stability of the DNA overlay. We generate a churn phase and create a snapshot within this phase. The real neighbors of a peer (as obtained form the global view offered by the simulation) are compared to the neighbors currently seen by the peer. In Figure 3 we show how many of its five direct neighbors a peer actually knows in dependence of the churn rate in the system. On average a peer knows more than 4.5 of its 5 direct neighbors even if the average peer stays online for only 30 minutes. Note that the correctness of a peer's neighbors does not depend on the size of the network at all. The curve progression is almost identical for 500 and 100 peers. That is, the degree of stability of the overlay network does not depend on the size but on the current churn rate of the system.

The above results show that the DNA overlay offers scalable search times and is robust against churn. The remaining question is how much bandwidth the DNAs need to maintain the overlay network. Figure 4 plots the average maintenance traffic of a single peer against the total number of peers in the overlay network. Again the lower curve represents a system without any churn. The larger the overlay network gets, the more neighbors are maintained and the more traffic is needed to keep these neighbors up-to-date. As can be seen in the figure the consumed bandwidth scales very well to the size of the system. The upper curve summarizes the same results for an average online time of 60 minutes. It has a similar progression, but illustrates that a peer uses more maintenance traffic during a churn phase.

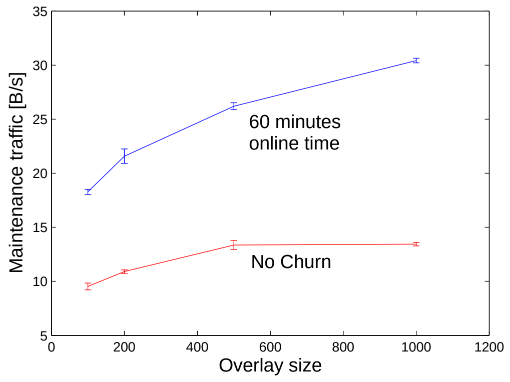

Fig. 4. Average maintenance traffic in dependence of the system size

To study the influence of churn on the bandwidth needed for maintenance in more detail, we did a parameter study for the churn rate in Figure 5. The average online time of a peer varies between 30 and 120 minutes. The shorter a peer stays online on average, i.e. the more churn there is in the system, the more maintenance traffic is produced by the DNA client. That is, the DNA adapts

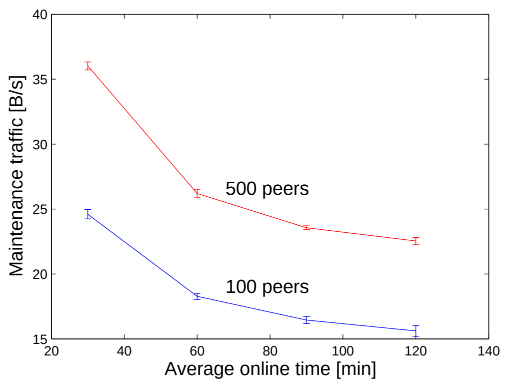

Fig. 5. Average maintenance traffic in dependence of the churn rate

automatically to the current churn rate. As stated above, the DNA needs more maintenance traffic in a larger network, as there are more neighbors that have to be kept up-to-date.

6 Conclusions and Future Work

In this paper we presented a novel technique for distributed fault and performance management. The proposed DNA framework is based on a self-organizing P2P overlay network (Kademlia) and offers plug and play functionality when integrating new DNA clients. The system is able to perform local tests on the client and distributed network tests in conjunction with other DNA clients. As a proof-of-concept, we built a running prototype and in addition proved its scalability by simulation. We investigated the robustness and reliability of the DNA in terms of churn behavior, i.e. the fluctuation of the size of the overlay network. A local test module, as well as examples for distributed tests were described in detail. The proposed distributed end-to-end architecture facilitates the provisioning and monitoring of new services offered by service providers.

Future work will be devoted to the integration of a new passive end-to-end QoS monitoring concept featuring performance management from the user point of view. The results of such test will be available to standard-network management systems via well-established interfaces, like SNMP traps or MIB variables. Thus, it can be seen as a QoS-enabling complement of existing network performance management solutions.

Acknowledgments

The authors would like to thank Phuoc Tran-Gia and Stefan Chevul for the helps and discussions during the course of this work. Additional thanks go to the EuroNGI FP6 Network of Excellence for supporting this work.

References

1. Subramanian, M.: Network Management – Principles and Practice. Addison-Wesley (2000)
2. M. Fiedler (ed.): EuroNGI Deliverable D.JRA.6.1.1 – State-of-the-art with regards to user-perceived Quality of Service and quality feedback (2004) URL: http://www.eurongi.org/, http://www.ats.tek.bth.se/eurongi/dwpjra611.pdf.
3. (HP OpenView Management Software) URL: http://www.openview.hp.com/.
4. (IBM Tivoli Software) URL: http://www.ibm.com/tivoli/.
5. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: IPTPS 2002, MIT Faculty Club, Cambridge, MA, USA (2002)
6. (Autonomic Communication (IST FP6 Project)) URL: http://www.autonomic-communication.org/.
7. Binzenhöfer, A., Tutschku, K., auf dem Graben, B.: DNA – A P2P-based Framework for Distributed Network Management. In: Peer-to-Peer-Systeme und -Anwendungen, GI/ITG Work-In-Progress Workshop in Cooperation with KiVS 2005, Kaiserslautern (2005)
8. Wawrzoniak, M., Peterson, L., Roscoe, T.: Sophia: an information plane for networked systems. In: SIGCOMM Comput. Commun. Rev. Volume 34., ACM Press (2004) 15–20
9. Chen, Y., Bindel, D., Song, H., Katz, R.H.: An algebraic approach to practical and scalable overlay network monitoring. In: SIGCOMM Comput. Commun. Rev. Volume 34., ACM Press (2004) 55–66
10. Fiedler, M., Tutschku, K., Carlsson, P., Nilsson, A.: Identification of performance degradation in IP networks using throughput statistics. In Charzinski, J., Lehnert, R., Tran Gia, P., eds.: Providing Quality of Service in Heterogeneous Environments. Proceedings of the 18th International Teletraffic Congress (ITC-18), Berlin, Germany (2003) 399–407
11. Josephson, W.K., Sirer, E.G., Schneider, F.B.: Peer-to-peer authentication with a distributed single sign-on service. In: The 3rd International Workshop on Peer-to-Peer Systems IPTPS'04, San Diego, USA (2004)
12. Xu, J., Kumar, A., Yu, X.: On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. In: IEEE Journal on Selected Areas in Communications. Volume 22. (2004)
13. Binzenhöfer, A., Tran-Gia, P.: Delay Analysis of a Chord-based Peer-to-Peer File-Sharing System. In: ATNAC 2004, Sydney, Australia (2004)
14. Binzenhöfer, A., Staehle, D., Henjes, R.: On the Stability of Chord-based P2P Systems. University of Würzburg, Technical Report No. 347 (2004)