DNA
A P2P-based Framework for Distributed Network Management

Andreas Binzenhöfer Kurt Tutschku
Björn auf dem Graben
{binzenhoefer, tutschku@informatik.uni-wuerzburg.de}

Abstract: Traditional network management has to cope with the disadvantages that come along with a central management unit. In this paper we present a framework for distributed network management using a p2p overlay network consisting of several Distributed Network Agents. The framework provides a reliable and scalable basis for distributed test, like e.g. the identification of performance degradation in IP networks using throughput statistics. The framework is not intended to replace the central network manager, but rather to support it in surveying the status of the corresponding network.

1 Introduction

IP networks have become highly complex in implementing firm Quality-of-Service (QoS) characteristics and high reliability features. In order to facilitate a cost efficient operation of these networks, the future control mechanisms have to be realized in a more *autonomous* way than in today's networks. The concept of *Automonic Computing (AC)* has recently gained tremendous attention in the industry [1, 2, 3]. Autonomic Computing is an approach towards self-managed computing systems with a minimum of human interference, that comprises four functional domains: self-configuration, self-optimization, self-healing, and self-protection. In this paper we take the concept of Autonomic Computing and extend the idea to *Autonomic Networks*, i.e. to autonomously operating networking systems such as LANs and WANs. The goal is to have an overlay network of *Distributed Network Agents (DNAs)* for distributed network management. The remainder of this paper is structured as follows. In Section 2 we give an answer to the question why a central device does not suffice for today's network management. Section 3 introduces the DNA framework and Section 4 presents some possible applications of our p2p-based framework. Section 5 finally concludes the paper and gives an outlook to future work.

2 Why not use a Central Network Manager?

When creating a framework for distributed network management the question of why we are not simply using a central network manager, like e.g. IBM's Tivoli or HP Openview, is inevitable. In this section we therefore summarize the three most important disadvantages

that come along with a central manager. We will show later that these problems can be solved by our framework.

The most obvious disadvantage of a central network manager is its property of a single point of failure. Once the single central monitoring unit and its possible redundant backup fail, 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.

Another disadvantage is the fact that network management based on a single unit does not scale to larger networks. 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.

The third disadvantage we would like to mention is the limited view of the central monitoring unit. While a central network manager is able to monitor, e.g., client A and the webserver, it has no means of knowing the current status of the connection between the monitored devices themselves. That is, the part of the network that can be monitored by a single unit has the shape of a star as shown in Figure 1. There is no way for the central unit to probe the state of the connection between the edges of this star. The status of the corresponding link remains unknown to the central manager.

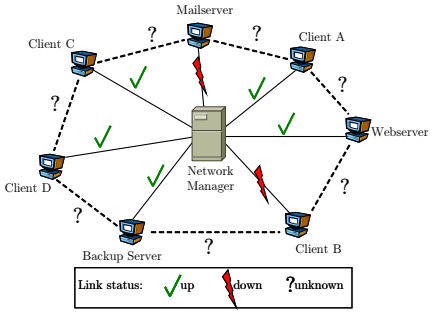

Figure 1: A simplified presentation of a network. The central network manager has a limited view on the surveyed network.

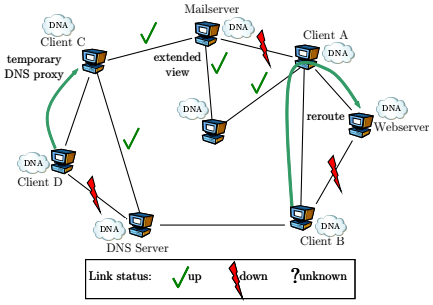

Figure 2: Clients running the DNA extend the view of the network and offer new applications like temporary rerouting or proxy functions.

3 The Distributed Network Agent

In this section we introduce our framework for distributed network management. The Distributed Network Agent (DNA) is a software client running on a host, a server or any other active component of a network. Since the majority of all reported network failures is indeed caused by local problems, the DNA is divided into two phases:

- Phase 1: Local Tests
- Phase 2: Distributed Tests

The two phases will be explained in detail in the following two subsections.

3.1 Phase 1: Local Tests

Before a DNA takes part in managing the network, it tries to rule out the possibility of local errors. This is done by a set of tests that are performed on the local host. These tests can be divided into three classes: Hardware, Local Configuration, and Network Connectivity. Possible tests concerning the hardware are, e.g., the verification of the status of the network interface card or the status of the physical connection to the network itself. Tests dealing with the local configuration include DNS and IP configuration, possible errors in the routing table, log files, and the like. The network connectivity can, e.g., be tested by pinging the local IP or some well known hosts. When the local integrity is ensured the DNA can join the overlay network in phase 2.

3.2 Phase 2: Distributed Tests

We use a virtual overlay network as a basis for our distributed framework. The main purpose of the overlay network is to keep the participating DNAs connected in such a way that each DNA has the possibility to find any other DNA in reasonable time. Thereby it is irrelevant, whether the underlying physical network is a wired or a wireless solution. It can even be a mixture of both. What is important, however, is the number of overlay connections a DNA has to maintain to keep the overlay connected and of course the time needed to find another random DNA. Both problems can be solved using a distributed hash table (DHT). Chord [7], Pastry [5], CAN [6] and Kademlia [8] are scalable P2P based realizations of a DHT that do not rely on a single point of failure. All of them are furthermore able to retrieve information from the DHT using $O(log(n))$ overlay hops, while only maintaining connections to $O(log(n))$ other peers in a network of size n. We chose Kademlia as a basis for our overlay network for numerous reasons that go beyond the scope of this paper. The important thing is, that we are able to locate other DNAs in reasonable time, while having a negligible risk of losing the structure of the overlay network. Moreover, instead of having to add a new component to the network by hand, new DNAs can automatically be added to the network in a plug and play manner using the Kademlia join algorithm. In the next section we discuss some possible applications of our framework.

4 Areas of application

Once the DNAs have joined the overlay network, they provide a feature rich basis for distributed network tests as shown in Figure 2. One possible application is a pinpoint module intended to locate faulty links or bottlenecks in the network by letting the DNAs ping each other. This could as well be done using throughput statistics as shown in [4]. Furthermore the DNAs can use each other as temporary proxies with different functionality. If

one DNA, e.g., loses its DNS server, it could ask another DNA to temporarily resolve its DNS-queries. It would even be possible to build a temporary routing table on application level to circumvent bottlenecks in the physical network or to establish load balancing. Due to the extended view of the network the DNAs could also be used for network tomography. To prove the feasibility of these applications we will integrate the corresponding modules into our prototype.

5 Conclusion

In this short paper we outlined the framework of the Distributed Network Agent (DNA) and sketched different possible ways of how to use this framework for distributed network management. We implemented a prototype of the DNA in .NET with a communication interface based on Kademlia. We realized local tests, as well as distributed ping tests, a bandwidth measurement tool and a temporary DNS proxy. The framework can easily be extended with additional test modules. In future work we will make measurements to evaluate the overhead needed to maintain the overlay network and prove the scalability and efficiency of our prototype by simulation.

Acknowledgments:

The authors would like to thank Markus Fiedler and his group from BTH for the helps and discussions during the course of this work. Additional thanks go to Datev for their cooperation and support in the DNA project.

References

[1] IBM Autonomic Computing: *http://www.research.ibm.com/autonomic/*

[2] SUN N1 Grid Solution: *http://wwws.sun.com/software/n1gridsystem/index.html*

[3] Microsoft Dynamic System Initiative: *http://www.microsoft.com/dsi*

[4] M. Fiedler, K. Tutschku and P. Carlsson, *Identification of Performance Degradation in IP Networks Using Throughput Statistics*, the 18th International Teletraffic Congress - ITC18, Berlin, Germany, September, 2003

[5] A. Rowstron and P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, November, 2001

[6] Sylvia Ratnasamy et. al, *A scalable content-addressable network*, In Proceedings of the 2001 ACM SIGCOMM Conference, San Diego, California, USA, August, 2001

[7] Ion Stoica et. al, *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*, In Proceedings of the 2001 ACM SIGCOMM Conference, San Diego, USA, August, 2001

[8] P. Maymounkov and D. Mazieres, *Kademlia: A peer-to-peer information system based on the xor metric*, In Proceedings of IPTPS02, Cambridge, USA, March, 2002