Supporting Peer-to-Peer Computing with FlexiNet

Thomas Fuhrmann
Institut für Telematik
Universität Karlsruhe (TH), Germany

Abstract

Formation of suitable overlay-network topologies that are able to reflect the structure of the underlying network-infrastructure, has rarely been addressed by peer-to-peer applications so far. Often, peer-to-peer protocols restrain to purely random formation of their overlay-network. This leads to a far from optimal performance of such peer-to-peer networks and ruthlessly wastes network resources.

In this paper, we describe a simple mechanism that uses programmable network technologies to improve the topology formation process of unstructured peer-to-peer networks. Being a network service, our mechanism does not require any modification of existing applications or computing systems. By that, it assists network operators with improving the performance of their network and relieves programmers from the burden of designing and implementing topology-aware peer-to-peer protocols.

Although we use the well-know Gnutella protocol to describe the mechanism of our proposed service, it applies to all kinds of unstructured global peer-to-peer computing applications.

Index terms — Overlay-network formation, Topology shaping, Resource management, Global storage, Gnutella, Programmable networks

1 Introduction

FlexiNet is a programmable network infrastructure currently spanning the German research network with *active nodes* located at three major German universities (Berlin, Karlsruhe, and Munich) and stubs in a few commercial provider networks. These active nodes run the AMnet node operating software [10, 9] that allows them to capture, process, and re-inject network-level packets as required by the various *service modules* that populate the active nodes. Besides the execution environment that hosts the service modules, each AMnet node provides means for resource management, node evaluation, and service relocation. Node evaluation distributively determines on which node(s) a requested service will be set up. Service relocation moves an already running service module to another node if either local resources are about to be depleted, or if a service needs to follow, e.g., a mobile end-device it is supporting.

Since the active nodes seamlessly integrate into standard IP networks, FlexiNet's active nodes need only be deployed sparsely. A typical FlexiNet improved site[1] will contain an active node as border router connecting the site to its Internet Service Provider (ISP). If that connection is realized redundantly, active nodes should act as border routers for all of these connections. These active border routers can be supplemented by a number of additional nodes that can be used to scale the performance of the programmable network. See also figure 1 for a typical scenario.

Uses of FlexiNet are manifold and grow almost daily since the test-bed has gone live recently. Typical FlexiNet services include

- *Multicast* support for non-native IP multicast networks. Here, an active node detects a second request for a stream that is already routed through that node. It then does not forward the request upstream to the server, but directly duplicates the stream's content for the downstream client. This has been demonstrated to drastically reduce the bandwidth required for Internet radio applications that use HTTP connections. — The same principle can also be applied to other network services, like anycast and concast.
- *Transcoding* either for adaptive bandwidth reduction or for extension of interoperability. E.g., one of the FlexiNet demo services adapts an MPEG 2 digital TV broadcast for a PDA. This is especially useful when combined with the multicast service described above, since here sender-based mechanisms cannot be applied.

[1]FlexiNet does not assume that other sites besides the local site are equipped with active nodes. Although, the more sites are FlexiNet-enhanced, the more services can benefit from the programmable infrastructure.

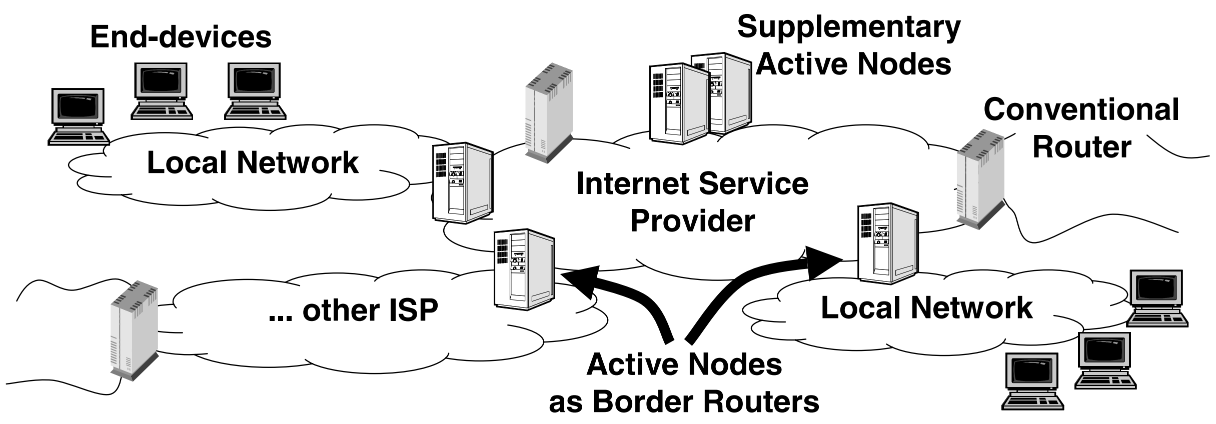
End-devices
Supplementary Active Nodes
Conventional Router
Local Network
Internet Service Provider
... other ISP
Local Network
Active Nodes as Border Routers

Figure 1. Typical setting for the FlexiNet infrastructure: An Internet Service Provider (ISP) employs FlexiNet's active nodes as border-routers both for its customers' local networks and for its peerings with other ISPs.

- *Mobility support* — Such services greatly benefit from the combined capability to interact with regular IP traffic (like e.g. a network address translator) while being able to quickly relocate to another active node if required (like e.g. agent-based software components).
- *Active security* mechanisms that detect and battle against attacks and intrusion attempts. — These services also draw from the flexibility to quickly set up services that match a certain requirement, e.g. block traffic in a distributed denial of service (DDoS) attack close to the originating site.

Such and other uses of FlexiNet and similar programmable networking architectures have already been described elsewhere [8]. In this paper, we will explain a new use of the FlexiNet infrastructure that can improve *peer-to-peer* computing.

We describe a service module for the FlexiNet infrastructure that equips topology-unaware P2P applications with a mechanism to construct an optimized overlay structure. Being a network service, applying this mechanism does not require modification of the P2P application or of the hosts that run these applications.

Although, in the following, we demonstrate the effectiveness of this service with the Gnutella protocol, the very same mechanism can be used to improve the topologies of all unstructured P2P networks. This helps network operators to enhance their networks' performance and embanks the consequences of uncontrolled deployment of resource-consuming P2P applications. Additionally, this mechanism can also serve P2P application designers as an easy method to make their applications topology aware, by shifting the task of topology detection from the end-device back into the network.

Considering these advantages, it can be expected that the application of the proposed mechanisms widely improves the performance of *peer-to-peer computing* in all application areas.

2 Service Module Overview

Peer-to-peer (P2P) computing comes in various flavors, extensions, and intensities. Napster, probably one of the most well-known P2P applications, in fact used a central server for locating the data and a genuine P2P approach only for downloading the data. Other classical Internet applications, like e.g. the Network News Transfer Protocol (NNTP) [14] already used typical P2P mechanisms for what was then called "host-to-host" communication. Similar mechanisms are, e.g. applied in the Gnutella system that allows users to locate and retrieve files from the random network formed by (all the) other Gnutella peers.

For demonstrating the use of our FlexiNet service we use the well-known Gnutella system [4, 15, 21, 23, 24] since it has been comparatively throughly studied. The same mechanism applies similarly to other unstructured P2P networks, like FreeNet [5] or Morpheus, KaZaA, and other FastTrack-based systems, that have reached wide-spread deployment especially among today's student generation. It can hence be expected that the service presented here can significantly reduce the network load of academic and commercial networks.

2.1 Short review of the Gnutella system

Each Gnutella node is assumed to be equipped with a (short) list of other Gnutella nodes, e.g. cached or well-known nodes. Upon start-up, a Gnutella node establishes a connection to one of these nodes, thereby becoming connected to the Gnutella overlay-network. In order to retrieve a file, a node floods its vicinity of the overlay with an according request. Here, "vicinity" means that part of the overlay that can be reached within a given number of (overlay-)hops. After receiving a positive response from another overlay-node, i.e. a peer, the requesting peer directly connects to the responding peer to retrieve the requested file. A second protocol mechanism, the *ping-pong* mechanism, similarly floods the node's vicinity to find more potential peers that it can connect to in order to increase its number of links.

2.2 Problem statement

By this mechanism, Gnutella creates a *random graph* with *power-law* structure which has some advantageous properties [2, 1] but comes also with various draw-backs, e.g. the lack of directed search [6], a mismatch of the overlay-topology with that of the underlying network [20, 27], and an astonishingly large bandwidth consumption [21, 13].

For our case, where we consider a stub-network considerably populated with overlay-nodes[2], bandwidth consumption and overlay-topology is important. Given the random nature of the Gnutella network, it is very unlikely that nodes from the local network peer with other local nodes. Rather, each node most likely establishes multiple connections with nodes spread all over the world. This leads to two entwined, bandwidth-consuming effects:

- Queries are flooded in the overlay-vicinity not in the nodes' actual vicinity. Given the fact that overlay-link utilization is independent from the node's position in the network [21], the required bandwidth grows linearly with the number of nodes in the local network, i.e. *no scale effects* are realized.

- Even worse, this topology mismatch leads to an extended bandwidth waste since files are likely to be retrieved from remote nodes rather than from nodes within the local network, although probability is high that there are plenty of coinciding requests from the local user population. However, the Gnutella protocol (as well as the other unstructured P2P protocols that are so popular with today's students) prevents the network from benefiting from this shared interest.

[2]E.g., a university network, including students' dormitories, where many students share their latest home-brew music or video-tapes from their favorite party or sport event.

Several authors have proposed improved protocols that better match the topologies of overlay-network and underlying infrastructure [22, 28, 20, 27]. Unlike the approach presented in this paper, these protocols need to be incorporated into the P2P application itself, either directly or via P2P libraries or middleware, like the JXTA system [11, 26]. As many networking examples have demonstrated in the past, this is an often unsurmountable barrier. Users tend to get locked into popular but inefficient applications and protocols, even if those protocols are just above the lowest limit of basic effectiveness.

The deployment of a network service, however, can remove this lock-in: It does not require the users to switch to another application using an improved protocol, but improves the network performance by transparently correcting some aspects of an inefficient protocol. We will demonstrate this with our Gnutella-improving FlexiNet service.

2.3 Supporting Peer-to-Peer Computing with FlexiNet

The main obstacle for topology-optimized overlay-construction with unstructured P2P applications like Gnutella is the lack of a mechanism to detect peers in the actual network's vicinity. An active node, however, has such knowledge since it sees all overlay-links leaving the site. We employ this knowledge by encouraging links to local peers and suppressing links to remote peers. By that, we distort the overlay-topology in favor of a local cluster of peers that are more loosely connected to the rest of the overlay network than they would have been without our intervention.

This approach removes both drawbacks listed above:

- Since the number of links to peers at remote sites is largely reduced, the amount of query-traffic is also reduced significantly.

- If a request *can* be satisfied locally chances are high that it actually *will* be satisfied locally. This is so much more important when we make the plausible assumption that common interests among the users of one network-site will lead to a high probability of local query hits.

In the following section we will explain how FlexiNet achieves the goal that has been sketched out so far.

3 Implementation

This mechanism is currently being implemented with FlexiNet [9]. This implementation can draw from the collection of service modules already developed for various other services in the FlexiNet testbed [8], e.g. the *TCP interceptor*. This module allows the monitoring of arbitrary TCP connections. If required, this module can even modify the content conveyed via the connection.

3.1 Key Implementation Elements

In our context the *TCP interceptor* enables us to

- Watch for new connections to the Gnutella port 6346.
- Scan new connections for the Gnutella request string (`GNUTELLA CONNECT`) in order to also notice connections to non-standard ports[3]
- Splice intercepted TCP connections that have been modified but do not need further modification[4]. (Splicing connections creates very low overhead on the active node.)
- Read *ping* and *pong* messages to gain an overview of both, the local topology and the remote peers.
- Suppress *pong* messages from potential remote peers.
- Insert *pong* messages that point to potential local peers.
- Insert *bye* messages to gracefully close an unwanted connection.

Based on these building blocks taken from the standard FlexiNet distribution, our new service does the following:

The first connection originating from or destined into the local network is monitored but otherwise left unmodified. All subsequent connections are — with increasing probability — softly blocked. By "softly blocking" we mean that the connection is accepted by the active node but no action is taken to actually connect to the requested node. Instead, *ping* messages on such a softly blocked connection will be answered with imitated *pong* messages that suggest local Gnutella nodes as potential peers to local nodes and accordingly remote nodes as potential peers to remote nodes. This reduces the number of connections that cross the active node and increases the coherence of the local Gnutella nodes without being directly noticed by these nodes. After some time or a certain amount of such *pong* messages, we send a *bye* message and close the connection again. By this, we limit the amount of state that needs to be held at the active node.

[3]It's up to the individual configuration if all newly opened TCP connections are scanned, or if one restrains to scanning only connections to the Gnutella port.

[4]Our current implementation of the *TCP interceptor* module treats scanning and modifying connections in the same way. Thus we need the splice there, too.

The probability for softly blocking a connection increases strongly with the number of connections to that particular node and weakly with the total number of Gnutella connections currently monitored. This maintains diversity and keeps the resilience properties of the Gnutella network while effectively restricting the actual number of connections linking the local site to the rest of the Gnutella overlay-network.

3.2 Preventing Network Separation

In fact, there is even a risk that under certain circumstances this mechanism might be over-effective. If local nodes rely on caching mechanisms only, i.e. if not from time to time knowledge of new remote peers is introduced into the system[5], the probability for the creation of new connections drops. If this creation rate drops below the rate with which peers leave the Gnutella network, the system is in danger of becoming disconnected.

We consider this risk only theoretical, but one should pay attention to it when using this service.

We propose the following countermeasure if this effect nevertheless becomes noticeable, e.g. because the local user population has become mature and no new nodes enter the Gnutella network while at the same time the outside population is highly volatile and nodes often vanish from the Gnutella overlay.

When the number of active connections that link the local and the remote part of the Gnutella overlay drops below a critical threshold, the FlexiNet node actively initiates new connections to prevent separation into two disconnected network parts. This is done by issuing a *Gnutella connect* both, into the local and into the remote part of the Gnutella network, i.e. the active node itself becomes a kind of a Gnutella node that offers no files and that is invisible to its two peers. The latter is achieved by not responding to *ping* messages and not modifying the *hop* and *TTL* counter when relaying messages.

If it becomes necessary to initiate several such connections to keep the local part of the Gnutella network well connected to the rest of the network, messages are *not* relayed across these different connections, i.e. the active node does not become a hub for Gnutella network. Thereby, the FlexiNet service does not attract traffic but simply always maintains a certain level of connectedness.

3.3 Application to Hierarchical Networks

The ability to enforce a certain topology structure in a P2P network that has not been designed to respect the struc-

[5]Under normal circumstances users will introduce plenty of such potential new remote peers by using information gathered outside the Gnutella network, e.g. from recommendations published in news groups, chat channels, or at web-sites.

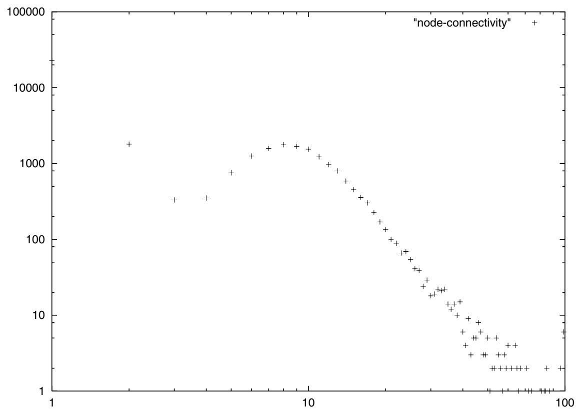

Figure 2. Node connectivity resulting from a typical simulation run. Plotted is the number of nodes (y-axis) connected by a given number of overlay-links (x-axis). The maximum number of links for this simulation run was set to 100.

ture of the underlying network, helps ISPs and other network operators to control use of network resources without completely blocking such uncooperative P2P applications.

A nice side-effect of the mechanism described here is that it also automatically applies to hierarchies of network regions. Assume that an ISP aggregates several customer sites. If this service is installed at the border routers between the customer premises and the ISP as well as between the ISP and the backbone, this mechanism will automatically create locality within the customers (if possible) and within the ISP.

4 Performance Analysis

Besides pursuing the implementation of our proposed mechanism to be able to test it within the FlexiNet infrastructure, we also simulated its behavior in order to evaluate our mechanism for a large network. This simulation needs to verify that the structure of the resulting Gnutella overlay network does not prevent its intended functionality. The benefit of our approach, namely the reduction of the Gnutella traffic crossing the FlexiNet active node, does not need to be verified since it is directly imposed by our mechanism. Here, we chose to reduce the link number by 90%.

Our simulation generates a random graph by successively adding nodes [7]. Each node is linked to a randomly picked second neighbor of a randomly picked node. After a first period, in which 15 000 nodes have been added to the network, the adding process is interrupted and all nodes are enriched with five additional links to randomly chosen second neighbors.[6] Then, the adding process (as described above) is resumed again. Altogether, 40 000 nodes are created. The maximum link number per node is 100.

Figure 2 shows a typical node connectivity distribution resulting from our topology generator. The double-logarithmic plot shows the number of nodes that are connected by a given number of overlay-links. The plot mirrors the structure of a ripened Gnutella network (cf. also [21]).

The reachability of nodes within this network was then explored with random walks: To this end, we measured the probability distribution of the length of non-self-crossing random walks with a fixed starting point. If the random walk leads into a highly inter-connected hub, the walk has a higher probability to end there, since the probability to hit an already visited node is relatively high in these regions. Accordingly, we expect an oscillation probability that reflects the wandering between these hubs. This expected structure (fig. 3 top) can be seen in the result from the simulated network.

If we now employ our proposed mechanism (local group of 200 nodes), we see the appearance of a second structure, determined by the local part of the network that we artificially kept from fully linking to the rest of the network (fig. 3 bottom). In the random-walk exploration, we see a second "oscillation" with a larger period. This larger period corresponds to leaving and re-entering the stub-network whose gateway-router was equipped with our mechanism. This reflects the structure we imposed by our mechanism that — as stated above — directly reduces the number of Gnutella overlay links run through the FlexiNet active node and thus the traffic volume.

However, when we measure the probability distribution for shortest path distance within these simulated networks, we see that the results are almost identical. Average shortest path lengths, even for local nodes, increase by less than one hop. Hence, our mechanism reduces the traffic volume (in this simulation by 90%) without disturbing the reachability of nodes within the Gnutella overlay network.

Although, these results are still only preliminary they demonstrate the principal effectiveness of our proposed mechanism.

5 Related Work

Mechanisms similar to the one employed here, have been implemented in various P2P systems, many of which aimed at providing a middle-ware for the construction of P2P applications. But to the best of our knowledge none of these systems consider the separation of the P2P protocol from

[6]This "ripening stage" mirrors the ripening of the Gnutella overlay network, where some nodes remain connected over a long time, while others join and leave in relatively short intervals.

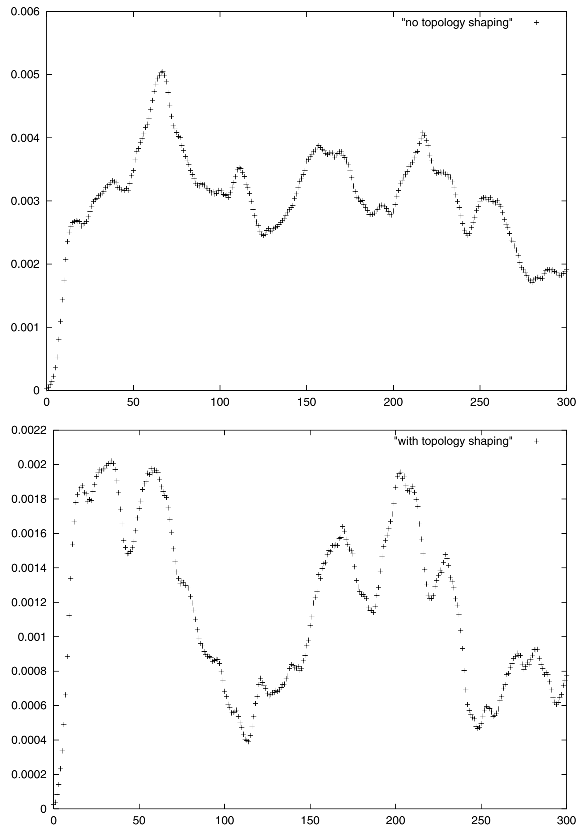

Figure 3. Topology structure for a network with (bottom) and without (top) topology shaping applied. The graph shows the measured node reachability explored with random walks. — Note that with topology shaping the average random-walk path length increases, thereby bringing the probability for short random-walk paths.

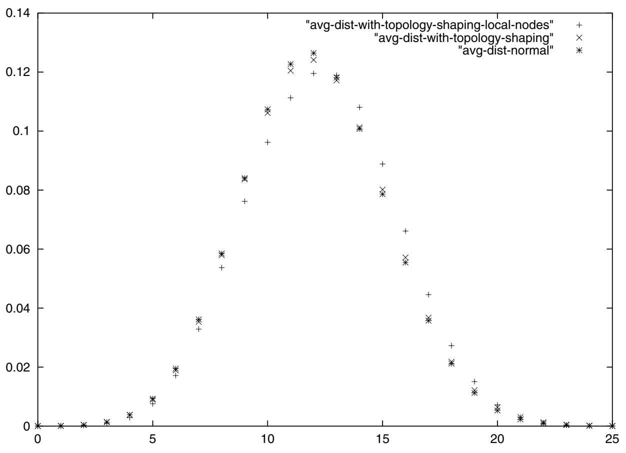

Figure 4. Probability distribution for shortest path distances with and without topology shaping applied.

the topology shaping as is done in the approach presented here.

[20] extends the original idea of a *content addressable network* (CAN) [19] by introducing a binning scheme that allows to match the topology of the overlay-network with that of the underlying network infrastructure. Pastry [22] and Tapestry [28] employ a routing scheme similar to radix tries, i.e. a longest shared prefix match. The desired topology match is achieved by selecting overlay-addresses according to some metric in the underlying network infrastructure. [27] proposes Mithos, an overlay-formation mechanism that is based on network delay measurements. For finding a close-to-optimal position for a node newly joining the overlay-network, Mithos successively probes neighboring nodes of the overlay. By following a path that shows a falling distance (measured in the underlying network-infrastructure) Mithos is likely to find an optimum, although it might get locked into a local optimum.

Compared to these approaches *programmable networks* can improve the performance of existing implementations and protocols. We consider this as a point of utmost importance since it empowers the site operators (or ISPs) to improve their networks without depending on their users (or customers) to switch to more efficient applications or protocols.

This particularly useful property is common to many programmable network approaches. Many such systems have been proposed during the recent years, e.g., the Click Modular Router [16], hierarchical service deployment [12], SILK [3], the Dynamically Extensible Router [17], and many others. These systems differ, e.g., in their hardware requirements, i.e. some use proprietary components, others a standard PC, like e.g. FlexiNet does. Some systems provide their own operating system [18, 25], others rely on a standard OS, e.g. the Linux kernel in the case of FlexiNet.

We believe that the mechanism we described in this paper is in no way specifically tied to FlexiNet and could be ported to many of such programmable network systems. Yet, we also believe that FlexiNet is very well suited for such a network service that supports global peer-to-peer computing by providing a powerful, secure, and scalable environment for such a service.

6 Conclusions and Outlook to Future Work

In this paper we have demonstrated how the FlexiNet programmable network can be used to improve Gnutella-like, i.e. unstructured, peer-to-peer protocols. Such protocols cover presumably most of the currently deployed P2P file-sharing and data-storage networks. Being a network service, the proposed mechanism does not require modification of any installed software or hardware component, but can straightforwardly be inserted into existing networks to improve their performance. Given the fact that globally deployed communication protocols otherwise tend to freeze in their first working version, we believe this to be a valuable means to promote the use of global peer-to-peer computing.

6.1 Extension Beyond P2P File-Sharing

Although we have demonstrated our proposed mechanism with the Gnutella file-sharing protocol, the very same mechanism easily applies to all unstructured P2P protocols, since it only affects the topology building aspects of the P2P application.

6.2 P2P Support beyond the Topology Formation Process

Besides the integration of support of other global P2P protocols besides Gnutella, there is another obvious extensions of the service described here, namely the interference with the actual content conveyed within the P2P overlay.

Again, there is no specific limit to the kind of interference, and we will and cannot elaborate on that topic. If the purpose of the P2P application is known to the service, all kinds of support might be given.

However, we can give a prominent example that applies to our context and that gives some idea of where mechanisms like the one proposed here can be extended to, namely the employment of copyright enforcement tools.

Such an application is an obvious consequence of the fact that monitoring the Gnutella messages also reveals query hits, as a consequence of which we can also monitor the subsequent file transfer. If the conveyed material can be recognized as violating copyright laws, we could block the transfer, corrupt the content or report this incident to the authorities.

However, we believe that this application of the FlexiNet system constitutes a completely different level of interference with the users' privacy rights, since then, we would no longer only modify signaling traffic, but actual content. While we consider the interference described in this paper as purely technical, since it just improves the functionality of a given communication system, the question of copyright versus privacy is different and should be discussed elsewhere. — As a last note, we would like to mention that we see a certain risk that an interaction that leads to a perceived service-degradation for the users is likely to be circumvented soon. Ultimately such circumventions could spoil the positive effects of mechanisms like the one described here.

References

[1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review E*, 64(046135), 2001.

[2] R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance in complex networks. *Nature*, 406(378), 2000.

[3] A. Bavier, T. Voigt, M. Wawrzoniak, L. Peterson, and P. Gunningberg. SILK: Scout paths in the Linux kernel. Technical Report 2002-009, Uppsala Universitet, Feb. 2002.

[4] F. R. A. Bordignon and G. H. Tolosa. Gnutella: Distributed sytem for information storage and searching — model description. rfc-gnutella.sourceforge.net.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2001.

[6] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of ACM SIGCOMM'02*, Aug. 2002.

[7] T. Fuhrmann. Experimental and mathematical protocol analysis — lecture notes. Institut für Telematik, Universität Karlsruhe, 2003. Available from http://www.tm.uka.de/lehre.

[8] T. Fuhrmann, T. Harbaum, P. Kassianidis, M. Schöller, and M. Zitterbart. Network services with FlexiNet. Available from http://www.flexinet.de.

[9] T. Fuhrmann, T. Harbaum, M. Schöller, and M. Zitterbart. AMnet 2.3 source code and programming documentation. Available from http://www.flexinet.de.

[10] T. Fuhrmann, T. Harbaum, M. Schöller, and M. Zitterbart. AMnet 2.0: An improved architecture for programmable networks. In *Proceedings of the Fourth Annual International Working Conference on Active Networks (IWAN'02)*, Zürich, Switzerland, Dec. 2002.

[11] L. Gong. JXTA: A network programming environment. *IEEE Internet Computing*, page 88, May/June 2001.

[12] R. Haas, P. Droz, and B. Stiller. A hierarchical mechanism for the scalable deployment of services over large programmable and heterogeneous networks. In *Proceedings of the International Conference on Communications (ICC'01)*, Helsinki, Finland, June 2001.

[13] S. Incorporated. Peer-to-peer file sharing — the effects of file sharing on a service provider's network, July 2002.

[14] B. Kantor and P. Lapsley. RFC 977: Network news transfer protocol, Feb. 1986.

[15] T. Klingberg and R. Manfredi. The gnutella protocol 0.6, June 2002. rfc-gnutella.sourceforge.net.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[17] F. Kuhns, J. DeHart, A. Kantawala, R. Keller, J. Lockwood, P. Pappu, D. Richards, D. Taylor, J. Parwatikar, E. Spitznagel, J. Turner, and K. Wong. Design of a high performance dynamically extensible router. In *Proceedings of the DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, May 2002.

[18] L. Peterson, Y. Gottlieb, M. Hibler, P. Tullmann, J. Lepreau, S. Schwab, H. Dandekar, A. Purtell, and J. Hartman. An os interface for active routers. *IEEE Journal on Selected Areas in Communications*, 19(3):473 –487, Mar. 2001.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM Conference*, Aug. 2001.

[20] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM*, 2002.

[21] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. In *IEEE Internet Computing Journal, 6(1)*, 2002.

[22] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01)*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[23] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN'02)*, San Jose, CA, USA, January 2002.

[24] R. Schollmeier and F. Hermann. Topology-analysis of pure peer-to-peer networks. In *Proceedings of Kommunikation in Verteilten Systemen (KiVS'03)*, Leipzig, Germany, Feb. 2003.

[25] N. Shalaby, L. Peterson, A. Bavier, Y. Gottlieb, S. Karlin, A. Nakao, X. Qie, T. Spalink, and M. Wawrzoniak. Extensible routers for active networks. In *Proceedings of the DARPA Active Networks Conference and Exposition*, pages 92–116, 2002.

[26] SUN Microsystems. *Project JXTA: Java Programmers Guide*, 2002.

[27] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. *ACM Computer Communication Review*, 33(1), Jan. 2003.

[28] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Apr. 2001.