NETWORKS AND HETEROGENEOUS MEDIA
©American Institute of Mathematical Sciences
Volume **3**, Number **2**, June **2008**

Website: http://aimSciences.org

pp. **251–265**

LOCALITY-AWARE P2P QUERY SEARCH WITH ISP COLLABORATION

Vinay Aggarwal

Deutsche Telekom Laboratories / TU Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Anja Feldmann

Deutsche Telekom Laboratories / TU Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Abstract. More than half of Internet traffic today is contributed by peer-to-peer (P2P) systems. Yet P2P systems build their overlay topology largely agnostic of the Internet underlay, which often leads to traffic management challenges for Internet Service Providers (ISP) and potentially inefficient neighbourhood selection for P2P nodes. To overcome this, we propose to use an oracle hosted by the ISPs, so that ISPs and P2P users can cooperate for improved performance. The oracle can be queried by P2P nodes while choosing neighbours for content search, and it will rank the possible neighbours of the querying node according to a locality indication, like location within the same AS, or the AS-hop distance. The ISP gains by keeping traffic within its Autonomous System (AS), and the P2P node can experience improved performance like lesser delay and better bandwidth.

In this paper, we evaluate the benefits of our scheme by performing experiments in a real Testlab as well as a simulation framework. We show how we configure representative AS topologies for P2P networks, and present experimental results with content search phase of a P2P network using different file sharing and search query distributions.

1. **Motivation.** A significant portion of the Internet traffic today is contributed by P2P systems, which are realized as overlays on top of the underlying Internet routing architecture. Some of the most popular applications that rely on the P2P methodology [10] include file sharing systems like Bittorrent, Gnutella and eDonkey, VoIP systems such as GoogleTalk and Skype, as well as real-time multimedia streaming and online gaming systems. Recent measurement studies have reported that P2P systems account for more than 50% of total Internet traffic [18, 19, 10].

Indeed, P2P system applications are one of the major reasons cited by users for upgrading their Internet access to broadband [15]. Yet, P2P traffic also poses a problem for the ISPs as it is very dominant and the traffic flows are almost impossible to engineer [18, 19]. This is primarily because ISPs have neither control nor knowledge of the overlay structure of P2P systems, which is formed largely independent of the Internet routing underlay.

2000 *Mathematics Subject Classification.* Primary: 68M10, 68M14; Secondary: 90B18.
Key words and phrases. P2P systems, query search, ISP-P2P collaboration, locality, Testlab.
The first author is supported by EU FP6 project DELIS..

As P2P systems often choose their neighbours without any respect for network locality, the traffic may have to cross ISP network boundaries multiple times [19, 2], even though the content is often served from within the ISP's network. Studies have shown that due to content language and geographical regions of interest, the desired content is often available "in the proximity" of interested users [8].

Let us consider how unstructured P2P file-sharing networks tend to form their topologies. New P2P nodes usually retrieve a list of members of the P2P network either via a well known Web page, a configuration file, or some history mechanism [17, 10]. They then pick some subset of these as possible neighbours either randomly [20] or based on some degree of performance measurement [9].

We, in [1], propose that each ISP offers a service, called the oracle, to its P2P users. When the P2P user wishes to connect to the overlay network, it supplies the oracle with a list of possible P2P neighbours. The oracle ranks them according to their proximity to the querying node. This ranking can then be used by the P2P node to select a closeby neighbour which improves the performance of the P2P system. The ISP can use this mechanism to steer P2P traffic, for example, to express preference for P2P neighbours that are within its network. The simplest version of such an oracle is a server that ranks IP addresses according to whether they are hosted by its ISP or not. The next better version may use AS-hop distance, which is defined as the number of AS-hops on the chosen BGP [16] route for the IP address, or take BGP policies into account. In addition, one may use intra-domain routing information, or characteristics like link bandwidth or customer service class to keep P2P traffic even more localized.

The benefits to P2P nodes are: (1) they do not have to measure the path performance themselves, as in [9]; (2) they can take advantage of the knowledge of the ISP; (3) they can expect improved performance in the sense of low latency and high throughput as bottlenecks, which are mostly found at inter-ISP transit/peering links [18], can be avoided.

The ISPs benefit as they can influence the neighbourhood selection process of the P2P network to, e.g., ensure locality of traffic flows and therefore again have the ability to manage the flow of their traffic. They also gain cost advantages, by reducing costs for traffic that leaves their network [18, 19].

While there are some proposals that aim to localize P2P traffic, e.g. [9, 3, 7], our solution is much simpler and more general. Besides being applicable to P2P nodes of all overlays, it promotes the idea of ISPs and P2P users actively co-operating such that both benefit, as discussed in detail in [1].

To evaluate the impact of using the oracle one should ideally study P2P systems with many nodes over the Internet, a network with many ASes and complex intra-AS topologies. Yet as the oracle is not yet offered by ISPs, it is not trivial to perform experiments on an Internet-wide scale. Hence, we have to make use of Testlab facilities and simulators for experiments. Simulations on the graph properties of oracle-influenced P2P overlays in [1] reveal that biased overlay graphs maintain a small diameter, small mean path length and node degree, while the densely connected subgraphs are now local to the ISPs. This implies that when P2P nodes consult an oracle while choosing their neighbouring nodes, they are able to keep a large number of their peerings within the AS, thus resulting in improved performance and lesser bottlenecks, while at the same time, do not sacrifice any of the nice graph properties of typical random graphs.

In order to be able to run actual P2P system code without having to model P2P networks and routing protocols, we use a Testlab facility to perform P2P experiments. The advantage of using a Testlab is that we can experiment with real traffic instead of simulated flows, and can configure network devices like routers, switches and links to generate variegated network scenarios and traffic environments. We get control over the network entities, which enables us to perform a wide range of experiments using real applications, network stacks and operating systems. Also, we have better control and visibility over the test environment as compared to running the experiments on the Internet, and can additionally eliminate the risk of inadvertently affecting the proper functioning of the Internet due to traffic generated by our experiments. Debugging and developing new applications hence becomes more feasible. To validate the Testlab results on much larger topologies, we later perform some large scale experiments in a simulation framework as well.

The paper is organized as follows. In Section 2, we introduce the hardware setup of our Testlab. We then explain how we configure various network topologies in the Testlab using routers, VLANs and other resources in Section 3. The experiments are performed with the Gnutella P2P file sharing protocol, which is briefly reviewed in Section 4. The actual Testlab experiments and results are detailed in Section 5, followed by a validation of these results on large-scale topologies in a simulation framework in Section 6. We finally conclude in Section 7.

2. Introduction to the testlab.

We use a **Testlab**, which is a collection of real devices - routers, switches and computing machines, that provides us the facility to perform experiments with real traffic in a realistic environment.

The hardware setup of the Testlab consists of the following devices:

- three Cisco 2691XM routers, named `c1,c2,c3`
- three Juniper M7i routers, named `j1,j2,j3`
- one Cisco 3750G24-TS switch, named `c4`
- three Cisco 2950SX-24 switches, named `c5,c6,j6`
- one Cisco 3500XL switch, named `j4`
- one Cisco 3550-12G switch, named `j5`
- nine Opteron-based load generator PCs, named `loadgen101` to `loadgen109`
- 13 Athlon-based load generator PCs, named `loadgen201` to `loadgen213`

A graphical representation of the setup is shown in Figure 1. The network is divided into four clouds: one cloud containing all Cisco routers, another cloud containing all Juniper routers, and two clouds of load-generators. The Cisco cloud and the Juniper cloud are connected to each other by the switches `c4` and `j4` and to the two load generator clouds by the switches `c5`, `j5`, `c6` and `j6`. Host-to-host connections can be set up either directly, using just a switch, or by using routed network links using one or more routers. All the PCs run Linux. We can design and setup different topologies on the Testlab hardware, in order to perform realistic experiments with multiple different scenarios.

3. Configuration of topologies in testlab.

To decide what kind of topologies we wish to configure in the Testlab, we need to consider how routing works in the Internet and in P2P systems. In the Internet, which is a collection of Autonomous Systems (AS), packets are forwarded along a path on a per-prefix basis. This choice of path via the routing system is heavily influenced by the contractual agreements between ASes [16] and the routing policy within the AS (usually shortest path

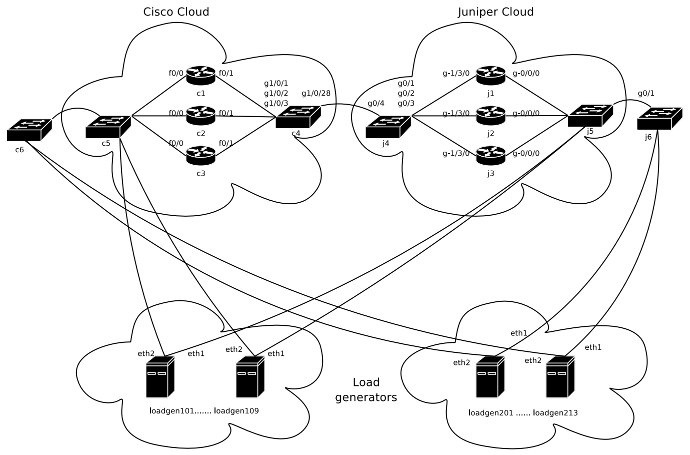

Figure 1. Hardware layout of Testlab

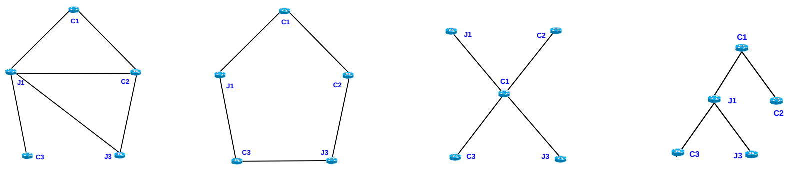

Figure 2. AS topologies used in experiments: realistic, ring, star and tree

routing based on a fixed per-link cost). P2P systems, on the other hand, set up an overlay topology and implement their own routing in the overlay topology which is no longer done on a per-prefix basis but rather on a query or key basis [10]. In unstructured P2P networks queries are disseminated, e.g., via flooding or random walks while structured P2P networks often use DHT-based routing systems to locate data. Answers can either be sent directly using the underlay routing or through the overlay network by retracing the query path.

Accordingly, we devise topologies with multiple ASes, where each AS hosts multiple machines running P2P applications. As a router can be taken as an abstraction of an AS boundary, and we have 5 routers available (one router was unavailable due to hardware malfunction), we decide to form 5-AS topologies. Each router connects to 3 load-generators, and we are able to run 3 instances of P2P applications on each machine concurrently. This gives us an upper bound of 5-AS topologies, with 15 machines, running 45 P2P clients concurrently. We connect the 5 routers in different fashions as shown in Figure 2, to arrive at 4 different AS topologies, which we call realistic, ring, star, and tree topologies. Running P2P experiments on different AS topologies enables us to analyse the impact of underlay topologies on P2P locality.

We now explain in brief how we configure the Testlab hardware to achieve the desired underlay topologies. All the interfaces are first assigned IP addresses using a pre-defined subnet layout structure. Since each router has only two interfaces, one for router-to-router connections and the other for router-to-loadgenerator connections, we have to assign multiple IP addresses to each router interface to create more than one router-to-router connections on a router. This is achieved by using *IEEE 802.1Q VLANs*.

Virtual LAN, commonly known as **VLAN** [26], is a group of devices on one or more LANs that are configured so that they can communicate as if they are attached to the same wire, when in fact they are located on a number of different LAN segments. Because VLANs are based on logical instead of physical connections, they are very flexible for user/host management, bandwidth allocation and resource optimization. By using VLAN-capable hardware devices, it is possible to define more than one Ethernet segments on a port-by-port basis without changing the hardware setup. In the Testlab we use the widely used IEEE 802.1Q [22] standard.

The router-to-loadgenerator connections are configured using the commands:

- **ifconfig**: for defining IP addresses on a particular ethernet interface
- **route**: for setting up the gateway of routes.

We thus configure each of the four different AS topologies shown in Figure 2 such that router-to-router connection is established by VLAN interfaces and each router is connected with 3 loadgenerators, which amounts to a total of 15 loadgenerators. The final configuration of the Testlab devices is shown in Figure 3. The configuration details of various topologies can be found in [5, 13].

4. Introduction to gnutella.

Having successfully configured a multiple-AS topology in the Testlab, we need to decide what P2P application to use for our experiments. Based on a number of considerations, we choose Gnutella [20]. One of the first decentralized overlay systems, Gnutella [20] is a popular file-sharing network with about two million users [8, 14]. Moreover, it is an open-source system with a well-known protocol, which has attracted a healthy interest from researchers, e.g., [2, 4]. Hence, its characteristics are well understood, and the existing literature allows us to compare and contrast our experimental results with established behaviour patterns of Gnutella, both in simulation frameworks as well as in the real Internet.

Each peer in the Gnutella network, called a *servent*, searches for other servents to connect to by flooding `Ping` messages, which are answered by `Pong` messages, that contain address and shared resource information. Each servent may or may not share any resources, and can search for content by flooding `Query` messages, which are answered by `QueryHit` messages. To limit flooding Gnutella uses TTL (time to live) and message IDs. Messages are generated with a TTL value of seven, and this value is decreased by one with each overlay hop that the message traverses. Messages are discarded when they reach a TTL value of zero. When a node receives a `Query` message, it checks if it has content that satisfies the query search string. If yes, the node generates a `QueryHit` message with a TTL value of the hops value of the corresponding `Query` plus two, lists content files that match the query string in this message, and sends it to the Gnutella node that originated the `Query` message. Each `QueryHit`/`Pong` message traverses the reverse path of the corresponding `Query`/`Ping` message. When a node receives multiple `QueryHit` messages for its search query, it selects one of the nodes randomly, and initiates a direct file download from this node using HTTP. Hence, while the negotiation traffic is carried within the set of connected Gnutella nodes, the actual data exchange of resources takes place directly between the relevant servents using HTTP, similar to other P2P protocols

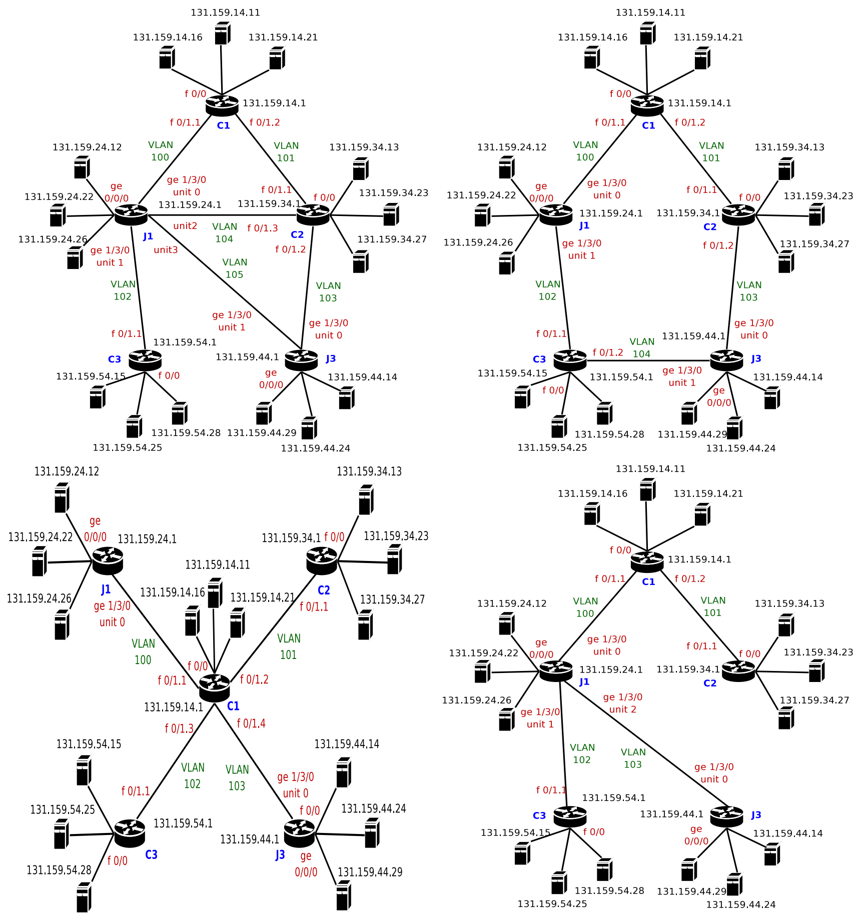

FIGURE 3. Configuration of Testlab devices to achieve the AS topologies: realistic, ring, star and tree

like Bittorrent. For scalability reasons, some servents are elevated to ultrapeers, while others become leaf nodes. Each leaf connects to a small number of ultrapeers, which in turn connect to a large number of both leafs and ultrapeers.

5. **Testlab experiments.** The first steps consist of installing Gnutella P2P software on each machine. To be able to install multiple Gnutella servents on each machine, we use the C-based GTK-Gnutella [25] software with a textual interface. By installing three servents each on 15 machines, we can have up to 45 Gnutella servents in our experiments. We designate one servent on each machine to be an ultrapeer, while the other two are made leaf nodes. Thus, we have 15 ultrapeers, and 30 leaf nodes.

5.1. **Realizing Biased Query Search.** A central machine which is connected to all the other load-generators is used to run the oracle. When a Gnutella servent sends a list of IP addresses to the oracle, the oracle sorts this list in the order of, first, servents within the querying servent AS, followed by servents in the AS which is one AS-hop away, followed by servents at increasing AS-hop distance.

To explore the aspect of content search and exchange using an oracle in an actual Testlab with real P2P traffic, we devise the following scheme. We first run an experiment with the unmodified Gnutella protocol running on each servent, which does not consult the oracle for neighbourhood selection. We then run another experiment, where each servent (both ultrapeer and leaf node) consults the oracle. To concentrate on content search and exchange, we let each servent communicate with the oracle and send the `Query` search messages initially to only those neighbours which are within its AS. If the search is unsuccessful, i.e., the servent does not receive a `QueryHit` within a predefined time interval, the servent sends the `Query` message to neighbours which belong to ASes that are one AS-hop away. If the `Query` is still unsuccessful, it is sent to all the remaining neighbours in the network. Hence, a biased Gnutella servent consults the oracle actively during the content search phase, but not during the bootstrapping phase. In contrast, servents in unmodified Gnutella flood the `Query` messages to all their connected neighbours.

As the file sharing pattern of P2P users can impact the content search experiment results, we employ two file sharing schemes:

- **Uniform**: every servent (both ultrapeer and leaf node) shares 6 unique files, leading to a total of 270 files in the Testlab.
- **Variable**: all ultrapeers share 12 files, half the leaf nodes share 6 files each, and the remaining leaf nodes share no files (free-riders). The content of files within any AS is kept the same as in uniform scheme, i.e., only the files of one leaf node within each AS are moved to its ultrapeer.

The aim of the experiments is to compare the impact of the oracle on the content search process of P2P systems. More specifically, we wish to compare the number of `QueryHit` messages received by each servent with and without consulting the oracle, for uniform and variable file sharing schemes. We let each servent introduce a unique search query string in the network. To better reflect P2P user behaviour, we use query strings that search for content of a particular type (e.g., mp3, rar), as well as those that search for something specific, e.g., an artist or album name [4].

5.2. **Results.** First, we measure the number of `Query` messages that are relayed in the entire Testlab network, and present the results in Tables 1 and 2. There are only 45 unique `Query` strings in both cases, but when a `Query` message is forwarded by a servent to its n neighbours, it is counted n times. This helps us to quantify the impact of biased neighbour selection on the scalability of the Gnutella network.

We see that consulting the oracle during content search reduces the number of `Query` messages that are relayed in the network, for both uniform and variable file sharing. The higher number of messages with variable file sharing is due to the fact that a `Query` often arrives at a servent which is not sharing any content, and is hence further forwarded to this servent's neighbours, thus generating more negotiation traffic. But even with variable file sharing, forwarding the `Query` messages with the help of ISP-hosted oracle to nearest neighbours reduces the negotiation traffic by at least 50%. As negotiation traffic for content search forms a significant portion of P2P traffic [4], we conclude that consulting the oracle significantly improves the scalability of such P2P networks.

We now measure the number of `QueryHit` messages received by each Gnutella servent, for the unique query string that it introduces in the network. We compare the number of responses received in unmodified Gnutella experiments with that of oracle-influenced Gnutella experiments. Figure 4 shows the results for uniform file

Topology	Unmodified P2P	Biased P2P
Realistic	6604	2473
Ring	6623	2512
Star	6679	2533
Tree	6643	2468

TABLE 1. Total number of Query search messages that are relayed in the network, using **Uniform File Sharing**

Topology	Unmodified P2P	Biased P2P
Realistic	10194	4873
Ring	10939	4834
Star	10902	4863
Tree	10872	4847

TABLE 2. Total number of Query search messages that are relayed in the network, using **Variable File Sharing**

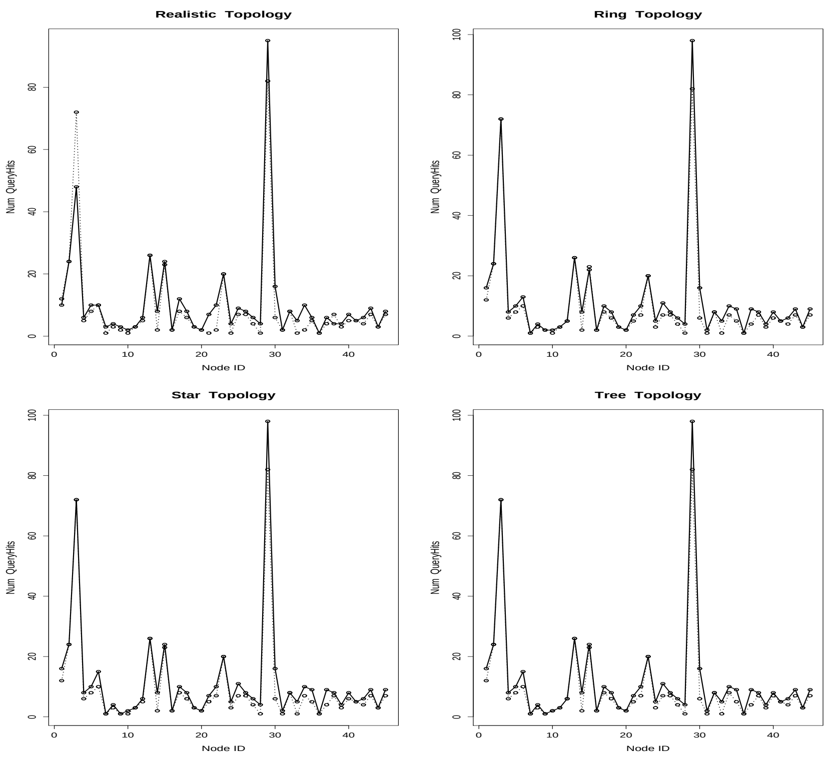

FIGURE 4. Plots showing the number of query response messages (y-axis) for each P2P node (x-axis) in the four topologies, for the case of **uniform file sharing**. The solid lines denote unmodified Gnutella, while the dotted lines denote the oracle-influenced Gnutella.

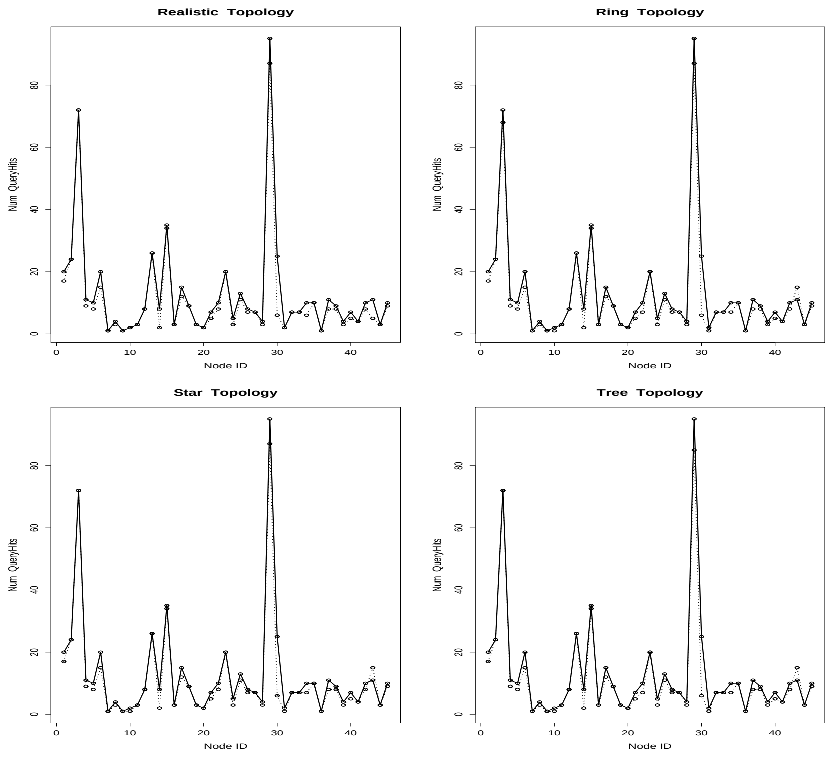

FIGURE 5. Plots showing the number of query response messages (y-axis) for each P2P node (x-axis) in the four topologies, for the case of **variable file sharing**. The solid lines denote unmodified Gnutella, while the dotted lines denote the oracle-influenced Gnutella protocol. The query strings used in these experiments are identical to those used in Figure 4.

sharing, while Figure 5 demonstrates the case for variable file sharing. The peaks in the plots correspond to general queries (e.g., mp3, rar), while the other values denote more specific content (e.g., artist or album name), reflecting their availability in the network. We see that while consulting the oracle during content search often reduces the number of `QueryHit` messages received by a servent, the difference is only nominal. Yet most importantly, we do not find a case where a `Query` yields a result in unmodified Gnutella, but fails to do so when consulting the oracle.

As the pattern and quality of query strings can also affect results [4], we run another set of experiments by changing the set of query strings in variable file sharing scheme. Here queries have a much lesser chance of finding file content, i.e., they are unlikely to yield a `QueryHit`. This helps us in detecting cases where servents get only a small number of `QueryHits` with unmodified Gnutella, but fail to yield any `QueryHits` at all when consulting the oracle.

The results are shown in Figure 6. We again see only a nominal reduction in the number of `QueryHit` messages for oracle-influenced Gnutella servents. Besides, we detect only 2 servents (both leaf nodes) from a total of 45, which did not receive any

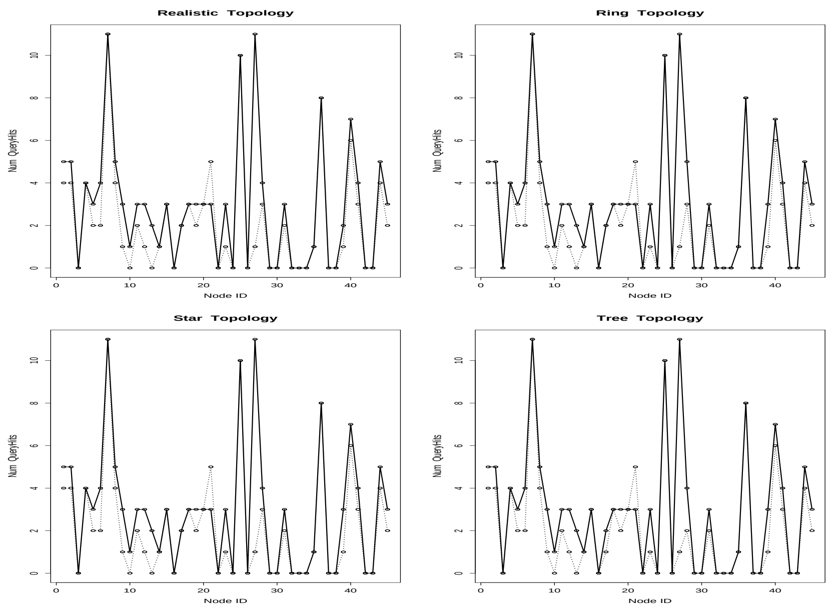

FIGURE 6. Plots showing the number of query response messages (y-axis) for each P2P node (x-axis) in the four topologies, for the case of **variable file sharing**. The solid lines denote unmodified Gnutella, while the dotted lines denote the oracle-influenced Gnutella protocol. The query strings used have a much lower chance of successful content search as compared to queries in Figure 4.

`QueryHit` when using the oracle, while they received 1 and 2 `QueryHit`s respectively with the unmodified Gnutella.

Hence, we conclude that consulting the oracle does not adversely affect the content search process of P2P networks. P2P nodes are easily able to search and share content, while the scalability of the P2P system improves considerably. The volume of P2P negotiation traffic in the network is reduced by at least 50%, while the content search performance remains comparable. Moreover, running experiments over different AS topologies as well as different file sharing models shows that the benefits accruing from consulting the oracle for P2P neighbour selection are independent of the underlay topology and P2P user behaviour.

6. **Experiments with a simulation framework.**

To validate the Testlab results in larger topologies involving hundreds of P2P nodes and to observe the effect of churn (nodes joining and leaving the network at short intervals) on P2P locality, we run larger-scale experiments in a simulation framework. We first introduce the simulation framework SSFNet, which we use to run experiments with a 700 node Gnutella topology. We then describe the simulation setup, followed by the results.

6.1. **Introduction to SSFNet.**

The Scalable Simulation Framework (SSF) [21] is an open-source discrete-event simulations standard for simulating large and complex networks. SSF Network Models (SSFNet) are Java models of different network entities, built to achieve realistic multi-protocol, multi-domain Internet modeling and simulation at and above the IP packet level of detail. These entities include Internet protocols like IP, TCP, UDP, BGP4, and OSPF, network elements like hosts, routers, links, and LANs, and their various support classes. Link layer and physical layer modeling can be provided in separate components. Domain Modeling Language (DML) is a public-domain standard for model configuration and attribute specification. It supports extensibility, inheritance and substitution of attributes. SSFNet models are self-configuring, i.e., each SSFNet class instance can autonomously configure and instantiate itself by querying network configuration files written in the DML format. A more comprehensive description of SSF can be found in [21].

We modify the neighbour selection procedure of Gnutella to take advantage of the oracle as follows [12]. Each Gnutella node sends the list of its potential neighbours it can connect to (typically stored in its Hostcache [20]), to the oracle. The oracle picks a node within the querying node's AS if it exists, or a random node otherwise. The node then establishes a Gnutella peering with this oracle-preferred node. This way, we influence the neighbourhood selection of P2P nodes at the bootstrapping stage, to choose a peer within the AS if it exists. Hence, while we use the oracle for biased query search in the Testlab, we use it here at the bootstrapping phase itself. We note that this neighbour selection mode is more restrictive as it discards inter-AS peerings as far as possible.

6.2. **Simulation setup.** The AS topology is derived from Internet measurements using the methodology explained in [1, 6]. The network consists of 1 level-1 AS, 5 level-2 ASes and 10 level-3 ASes. We place 355 nodes within the level-1 AS, 23 nodes within each level-2 AS, and 23 nodes within each level-3 AS. Within each AS, all the nodes are connected in a star topology to an intra-AS router. Each node in level-1 AS has a 1 Gbit network interface, each node in level-2 AS has a 100 Mbit network interface, while each node in level-3 AS has a 10 Mbit network interface. The links between level-l and level-2 ASes have a delay of 2 ms, while the links between level-2 and level-3 ASes have a delay of 10 ms. Each AS has 2 routers, one for the intra-AS node connections, and one for the inter-AS connections between different ASes. Thus, we have a topology with 16 ASes, 32 routers and 700 nodes running the Gnutella protocol.

As churn is a major characteristic of P2P applications, and most recent studies agree that online session length of P2P nodes is a heavy-tailed distribution [11], we model it using a Weibull distribution. To take into account the high amount of free-riding [14] in file-sharing systems, we model the number of files shared by each Gnutella servent as a Pareto distribution. Hence, a majority of the nodes share very little or no content at all, while a handful of servents offer most of the content in the system. Besides, we use query search strings that search for content of a particular type (e.g., mp3, avi, rap), as well as those that search for something specific, e.g., an artist or album name [4]. Not only do the P2P users share music content, but also video, software distributions and pictures. In this way, we reflect observed P2P user behaviour [11, 14, 4] realistically in our simulation framework.

Using the simulation setup explained above, we run two sets of experiments for $10,000$ seconds of simulation time each. We run one experiment where each P2P node follows the Gnutella protocol without consulting the oracle (unmodified P2P), while in the second case, each servent consults the oracle to choose neighbours within its AS if such nodes exist (biased P2P). We now present results to compare the two schemes.

6.3. **Results.** In order to validate the Testlab results, we first measure the number of query search messages relayed in the network, using unmodified as well as biased P2P networks. In each case, a total of about 900 unique query messages are generated by different nodes in the network, which are then relayed by the originating nodes to their connected neighbours. The total number of relayed query messages, observed at each time-to-live (TTL) value are shown in Table 3.

TTL	**Unmodified P2P**	**Biased P2P**
7	19,725	11,149
6	414,718	186,473
5	3,611,604	986,261
4	7,190,754	2,287,036
3	947,035	1,592,910
2	30,653	497,464
1	2,093	74,460
Total	12,216,582	5,635,753

Table 3. Total number of Query search messages that are relayed in the network

We observe that the number of query messages reduces from 12.2 million in the unmodified P2P network, to 5.6 million messages in the biased P2P network. This is a reduction of 54%. We also observe that consulting the oracle benefits the swarming pattern of query searches. From Table 3, we see that not only do the total number of flooded messages go down, rather, the reachability of queries at remote locations of the network increases as well. For example, the biased P2P network shows a much larger number of flooded query messages at TTL values of 1, 2 or 3, thus implying that queries are able to reach more P2P nodes at 5, 6 or 7 overlay hops from the originating node. This implies a more efficient swarming of search queries in the P2P network when nodes consult the oracle while choosing neighbours.

Table 4 shows the number of query response messages at different TTL values. We observe that the total number of query responses decreases only by 9.6%, a desirable feature as we naturally do not wish to obtain a lesser number of responses for queries when consulting the oracle. Figure 7 displays the logarithm of the number of search queries and their responses for both cases as a bar plot.

We also measure the impact of using the oracle on the quantity of network discovery traffic, i.e., number of `Ping` and `Pong` messages relayed in the network, see Table 5. Once again, we note a reduction in network discovery traffic by 42%, which translates into improved scalability of the P2P system.

While it is certainly desirable to improve the scalability of the P2P network, it is even more important to verify that consulting the oracle does not have a negative impact on the content search phase in a P2P network. In other words, it is important to analyse if the number of responses per search query are not adversely affected when P2P nodes bias their neighbourhood selection by consulting the ISP-hosted oracle.

Therefore we now compare the number of unsuccessful queries in unmodified and biased P2P networks. We find that while 24.78% queries do not find any content in the unmodified P2P network, 23.95% queries meet the same fate in the biased P2P

TTL	Unmodified P2P	Biased P2P
8	5	152
7	26	1,941
6	363	11,284
5	8,789	34,031
4	67,381	58,488
3	94,392	67,651
2	97,305	69,003
1	41	16
0	22	10
Total	268,324	242,576

TABLE 4. Total number of query search response messages that are relayed in the network

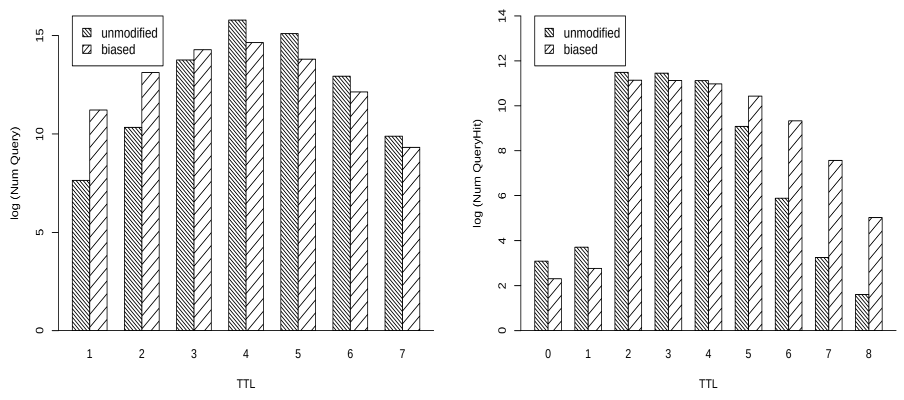

FIGURE 7. Bar plots to compare the logarithm of number of Query search messages (left), and query response messages (right), relayed in the network for unmodified P2P and biased P2P cases.

Message	Unmodified P2P	Biased P2P
Ping	15,323,903	8,986,961
Pong	153,021,689	89,491,751

TABLE 5. Total amount of network discovery traffic that is relayed in the network

network. Hence, we conclude that consulting the oracle does not affect the number of queries that do not find any content in the P2P network.

Finally, we compare the number of responses per query, for all satisfied queries in both the networks, and display it as a box plot [27] and cumulated density function (CDF) plot in Figure 8. We see that the number of responses per query exhibit similar distributions for both unmodified as well as biased P2P networks. The mean number of responses is 127.7 for the unmodified network, against 102.3 responses for the biased network. The median number of responses is 78 and 62 respectively.

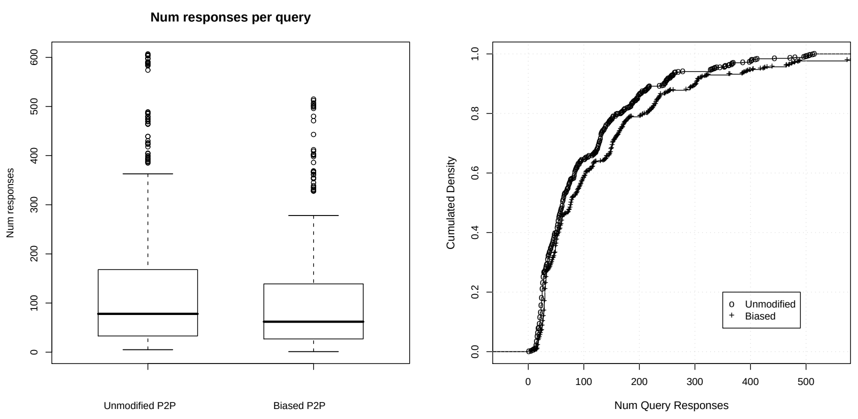

FIGURE 8. Figure showing a box plot (left) and CDF plot (right), to compare the number of responses per search query, for unmodified P2P and biased P2P cases.

While the average number of responses per query drops slightly while consulting the oracle, we note that the number of queries which do not match any content does not increase. Besides, the swarming pattern of queries improves considerably, the observed query responses are located at lesser overlay hops, and the scalability of the P2P network improves considerably.

7. **Conclusion.**

To evaluate the concept of P2P nodes consulting an ISP-hosted oracle for neighbourhood selection at query routing and bootstrapping stages, we perform experiments in a real Testlab as well as a simulation framework. We find that consulting the oracle reduces the negotiation traffic in the network by at least 50%, and improves the scalability of the P2P network considerably. This means that the overlay-induced traffic at the underlay is reduced by 50%. While there is a nominal decrease in the number of response messages to search queries, this does not adversely affect the search process, as all queries are still able to find content. Even with queries having a very low chance of success, cases of nodes unable to locate content due to consulting the oracle are very rare, if at all. Overall, we find that the P2P system continues to behave as per its protocol, with users able to locate and share content. The bottleneck links, which are present at AS access points, continue to route traffic without packet loss and congestion. Experiments with different underlay topologies, file distributions and query string patterns reveal that the benefits accruing from ISP-P2P collaboration are independent of these factors.

As a next step, we are experimenting with the oracle scheme in Planetlab to increase the scale of our experiments and to test the interaction of modified P2P clients with unmodified ones. We have realized the oracle as a Web server, and are in the process of installing Gnutella and Bittorrent clients on Planetlab nodes, which will consult the oracle while making neighbourhood selection decisions.

Acknowledgements. Thanks are due to Ruediger Wolf-Sebottendorf and Shailesh Gupta, for helping us with setting up the Testlab and performing initial experiments. We are greatful to the technical support staff at TU Munich, for providing us access to the Testlab setup. Our thanks also go to the anonymous reviewers for their valuable feedback.

REFERENCES

[1] V. Aggarwal, A. Feldmann and C. Scheideler, *Can ISPs and P2P systems cooperate for improved performance?* ACM SIGCOMM Computer Communications Review, **37** (2007), 31-40.

[2] V. Aggarwal, S. Bender, A. Feldmann and A. Wichmann, *Methodology for estimating network distances of gnutella neighbours*, in "GI Jahrestagung - Informatik 2004," 2004.

[3] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates and A. Zhang, *Improving traffic locality in bittorrent via biased neighbour selection*, in "IEEE ICDCS", 2006.

[4] A. Gish, Y. Shavitt and T. Tankel, *Geographical statistics and characteristics of P2P query strings*, in "IPTPS", 2007.

[5] S. Gupta, "Preparation of Testlab and Experiments with Selective Neighbour Selection for P2P Systems," Master Thesis, IIT Kharagpur and TU Munich, 2006.

[6] W. Muehlbauer, A. Feldmann, O. Maennel, M. Roughan and S. Uhlig, *Building an AS-topology model that captures route diversity*, in "ACM SIGCOMM", 2006.

[7] A. Nakao, L. Peterson and A. Bavier, *A routing underlay for overlay networks*, in "ACM SIGCOMM", 2003.

[8] A. Rasti, D. Stutzbach and R. Rejaie, *On the long-term evolution of the two-tier gnutella overlay*, in "IEEE Global Internet", 2006.

[9] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, *Topologically aware overlay construction and server selection*, in "IEEE INFOCOM", 2002.

[10] R. Steinmetz and K. Wehrle, "P2P Systems and Applications," Springer LNCS, 2005.

[11] D. Stutzbach and R. Rejaie, *Understanding churn in P2P networks*, in "ACM IMC", 2006.

[12] R. Tashev, "Experimenting with Neighbour Discovery Schemes for P2P Networks in a Simulation Framework," Master thesis, Technical University of Munich, October 2006.

[13] R. Wolf-Sebottendorf, "Building Complex P2P Topologies in a Testlab Environment," Master Project, Technical University of Munich, February 2006.

[14] S. Zhao, D. Stutzbach and R. Rejaie, *Characterizing files in the modern gnutella network: A measurement study*, in "MMCN", 2006.

[15] T. Mennecke, DSL Broadband Providers Perform Balancing Act, Online: `http://www.slyck.com/news.php?story=973`

[16] Y. Rekhter and T. Li, Border Gateway Protocol, Online: `http://www.ietf.org/rfc/rfc1771.txt`

[17] Slyck, Online: `http://www.slyck.com`

[18] CacheLogic Research, Online: `http://www.cachelogic.com/home/pages/research`

[19] Light Reading: Controlling P2P Traffic, Online: `http://www.lightreading.com`, December 2003.

[20] Gnutella v0.6 RFC, Online: `http://en.wikipedia.org/wiki/Gnutella`

[21] SSFNet, Online: `http://www.ssfnet.org`

[22] IEEE Computer Society: 802.1Q IEEE Standards for local and metropolitan area networks - Virtual Bridged LANs, Online: `http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf`

[23] Network Directory - Virtual LAN and IEEE 802.1Q, Online: http://www.networkdictionary.com/protocols/vlan.php

[24] Javvin. Network management and security, Online: `http://www.javvin.com/protocolVTP.html`

[25] GTK-Gnutella, Online: `http://gtk-gnutella.sourceforge.net`

[26] Wikipedia: Virtual LAN, Online: `http://en.wikipedia.org/wiki/Vlan`

[27] Box plot, Online: `http://en.wikipedia.org/wiki/Boxplot`

Received August 2007; revised November 2007.

E-mail address: `vinay.aggarwal@telekom.de`
E-mail address: `anja.feldmann@telekom.de`