Public Review for

Can ISPs and P2P Users Cooperate for Improved Performance?

Vinay Aggarwal, Anja Feldmann, and Christian Scheideler

This paper addresses the antagonistic relationship between overlay/p2p networks and IPS providers: they both try to manage and control traffic at different level and with different goals, but in a way that inevitably leads to overlapping, duplicated, and conflicting behavior. The creation of a p2p network and the routing at the p2p layer are ultimately treading on the routing functions of ISPs. The paper proposes a solution to develop a synergistic relationship between p2p and ISPs: ISPs maintain an "oracle" to help p2p networks in making better choices in picking neighboring nodes. The solution provides benefits to both parties. ISPs become able to influence the p2p decisions, and ultimately the amount of traffic that flows in and out of their network, while p2p networks get performance information for "free." The reviewers find that the problem is important and the solution is interesting and shows promise. An advantage of the method is that ISPs do not run into legal issues, since they do not engage in caching of potentially illegal content, they just provide performance information.

Public review written by
Michalis Faloutsos
University of California, Riverside

acm sigcomm

Can ISPs and P2P Users Cooperate for Improved Performance?

Vinay Aggarwal, Anja Feldmann
Deutsche Telekom Laboratories/TU Berlin
Berlin, Germany
{Vinay.Aggarwal, Anja.Feldmann}@telekom.de

Christian Scheideler
TU München
Munich, Germany
scheideler@in.tum.de

ABSTRACT

Peer-to-peer (P2P) systems, which are realized as overlays on top of the underlying Internet routing architecture, contribute a significant portion of today's Internet traffic. While the P2P users are a good source of revenue for the Internet Service Providers (ISPs), the immense P2P traffic also poses a significant traffic engineering challenge to the ISPs. This is because P2P systems either implement their own routing in the overlay topology or may use a P2P routing underlay [1], both of which are largely independent of the Internet routing, and thus impedes the ISP's traffic engineering capabilities. On the other hand, P2P users are primarily interested in finding their desired content quickly, with good performance. But as the P2P system has no access to the underlying network, it either has to measure the path performance itself or build its overlay topology agnostic of the underlay. This situation is disadvantageous for both the ISPs and the P2P users.

To overcome this, we propose and evaluate the feasibility of a solution where the ISP offers an "oracle" to the P2P users. When the P2P user supplies the oracle with a list of possible P2P neighbors, the oracle ranks them according to certain criteria, like their proximity to the user or higher bandwidth links. This can be used by the P2P user to choose appropriate neighbors, and therefore improve its performance. The ISP can use this mechanism to better manage the immense P2P traffic, e.g., to keep it inside its network, or to direct it along a desired path. The improved network utilization will also enable the ISP to provide better service to its customers.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: [Network topology]; C.2.4 [**Distributed Systems**]: [Distributed applications]

General Terms

Design, Experimentation, Management, Performance

Keywords

P2P, ISP, cooperation, routing, biased neighbor selection

1. INTRODUCTION

P2P systems have recently gained a lot of attention from the Internet users and the research community. Popular applications that use P2P systems include file sharing systems such as Bit-torrent, eDonkey, Kazaa, Gnutella as well as VoIP systems such as Skype and GoogleTalk [2]. P2P systems are so popular that they contribute more than 50% to the overall network traffic [3, 4, 5].

However, the wide-spread use of such P2P systems has put ISPs in a dilemma! On the one hand, P2P system applications have resulted in an increase in revenue for ISPs, as they are one of the major reasons cited by Internet users for upgrading their Internet access to broadband [6]. On the other hand, ISPs find that P2P traffic poses a significant traffic engineering challenge [4, 7]. P2P traffic often starves other applications like Web traffic of bandwidth [8], and swamps the ISP network. This is because most P2P systems rely on application layer routing based on an overlay topology on top of the Internet, which is largely independent of the Internet routing and topology [9].

To construct an overlay topology, unstructured P2P networks usually employ an arbitrary neighbor selection procedure [5]. This can result in a situation where a node in Frankfurt downloads a large content file from a node in Sydney, while the same information may be available at a node in Berlin. It has been shown that P2P traffic often crosses network boundaries multiple times [9, 10]. This is not necessarily optimal as most network bottlenecks in the Internet are assumed to be either in the access network or on the links between ISPs, but not in the backbones of the ISPs [11]. Besides, studies have shown that the desired content is often available "in the proximity" of interested users [10, 12]. This is due to content language and geographical regions of interest. Since a P2P user is primarily interested in finding his desired content quickly with good performance, we believe that increasing the locality of P2P traffic will benefit both ISPs and P2P users.

To better understand the origin of the problem of overlay-underlay routing clash, let us consider how routing works in the Internet and P2P systems. In the Internet, which is a collection of Autonomous Systems (ASes), packets are forwarded along a path on a per-prefix basis. This choice of path via the routing system is limited by the contractual agreements between ASes and the routing policy within the AS (usually shortest path routing based on a fixed per link cost) [13].

P2P systems, on the other hand, setup an overlay topology and implement their own routing [14] in the overlay topology which is no longer done on a per-prefix basis but rather on a query or key basis. In unstructured P2P networks queries are disseminated, e.g., via flooding [15] or random walks while structured P2P networks often use DHT-based routing systems to locate data [5]. Answers can either be sent directly using the underlay routing [5] or through the overlay network by retracing the query path [15]. By routing through the overlay of P2P nodes, P2P systems hope to use paths with better performance than those available via the Internet [14, 16]. But the benefits of redirecting traffic on an alternative path, e.g., one with larger available bandwidth or lower delay, are not necessarily obvious. While the performance of the P2P system may temporarily improve, the available bandwidth of the newly chosen path will deteriorate due to the traffic added to this path. The ISP then has to redirect some traffic so that other applications using this path receive enough bandwidth. In other words, P2P systems reinvent and reimplement a routing system whose dynamics should be able to interact with the dynamics of the Internet routing [7, 17]. While a routing underlay as proposed by Nakao et al. [1] can reduce the work duplications it cannot by itself overcome the interaction problems. Consider a situation where a P2P system imposes a lot of

traffic on an ISP network. This may cause the ISP to change some routing metrics and therefore some paths (at the routing layer) in order to improve its network utilization. This can however cause a change of routes (at the application layer) by the P2P system, which may again trigger a response by the ISP, and so on. In summary, we identify the following drawbacks:

- The ISP has limited ability to manage its traffic and therefore incurs potentially increased costs for its interdomain traffic, as well as for its inability to do traffic engineering on its internal network.
- The P2P system has limited ability to pick an optimal overlay topology and therefore provide optimal performance to its users, as it has no prior knowledge of the underlying Internet topology. It therefore has to either disregard or reverse engineer it.
- Different P2P systems have to measure the path performance independently.

While we do not know of a P2P network that tries to reverse-engineer the Internet topology, there are some proposals that suggest that P2P networks should bias their overlay topology by choosing neighbors that are close in the sense of high throughput or low latency, e.g., [18, 19, 20] or that are within the same AS, e.g., [10, 21]. Others such as the Brocade [22] system propose to build an overlay on top of a structured DHT P2P system that exploits knowledge of the underlying network characteristics. Yet another system [8] proposes to use caching to relieve the tension between ISPs and P2P systems.

We, in this paper, propose and evaluate the feasibility of a simpler solution where ISPs help P2P systems by offering an **oracle service**. The oracle acts like an abstract routing underlay to the overlay network but as it is a service offered by the ISP it has direct access to the relevant information and does not have to infer or measure it. For example, an ISP knows whether a customer has a DSL broadband or a modem connection, its link delay, etc. The benefit to the ISP is twofold: first, it can now influence the P2P routing decisions via the oracle and so regain its ability to perform traffic engineering (control the traffic flow) and second, the P2P measurement traffic to infer network distances is omitted. The P2P users benefit as explained below.

1.1 An oracle service

Let's consider how unstructured P2P networks tend to maintain 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 [2, 5]. They then pick some subset of these as possible neighbors either randomly [15] or based on some degree of performance measurement [18]. If the chosen neighbor cannot serve the new node it might redirect the new node by supplying an alternative list of P2P members.

Instead of the P2P node choosing neighbors independently, the ISP can offer a service, which we call the *oracle*, that ranks the potential neighbors according to certain metrics. This ranking can be seen as the ISP expressing preference for certain P2P neighbors. Possible coarse-grained distance metrics are:

- inside/outside of the AS
- number of AS hops according to the BGP path [13]
- distance to the edge of the AS according to the IGP metric [13]

For P2P nodes within the AS the oracle may further rank the nodes according to:

- geographical information such as: same point of presence (PoP), same city
- performance information such as: expected delay, bandwidth
- link congestion (traffic engineering)

This ranking can then be used by the P2P node to select a closeby neighbor although there is no obligation.

The benefit to P2P nodes of all overlays is multifold: (1) they do not have to measure the path performance themselves; (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 [11] can be avoided. That P2P networks benefit by increasing traffic locality has also been shown by Bindal et. al [21] for the case of BitTorrent.

The benefit to the ISPs is that they can influence the neighborhood 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. This will also allow them to provide better service to their customers and ensure fairness for other applications like Web traffic, etc. Besides, the ISPs will gain cost advantages, by reducing costs for traffic that leaves their internal network.

As the ability to control/manage its traffic is crucial to the operating costs of every ISP, we expect that the benefit accruing from this ability will outweigh the potential risks of providing an oracle, namely that the oracle exposes some information about the ISP topology and the network performance. As the oracle server only needs to roughly rank the IP nodes, it does not need to reveal more information about its network than can anyhow be inferred by reverse-engineering the ISP network via measurements [23].

The oracle is available to *all* overlay networks. One does neither need nor want to use a separate oracle for each P2P network. Furthermore, as an open service, it can be queried by any application and is not limited to file-sharing systems. Hence, querying the oracle does not necessarily imply participation in file sharing systems. This should limit the desirability of the oracle logs to, e.g., the music industry. Moreover the P2P system could permute, e.g., the last byte of the IP addresses it is interested in or use an anonymization service for querying the oracle.

Realizing an oracle service:
It may seem rather challenging to build such an oracle in a scalable manner, but much more complicated services, e.g., DNS, already exist. The oracle service can be realized as a set of replicated servers within each ISP that can be queried using a UDP-based protocol or run as a Web service. It can rely on a semi-static database with the ISP's prefix and topology information. Updating this information should not impose any major overhead on the ISP.

While the oracle service is not yet offered by the ISPs, P2P nodes have the chance of using a simple service to gain some of the oracle benefits already using the "pWhoIs" service [24]. This service is capable of satisfying 100,000 queries using standard PC-hardware [25] in less than one minute. It enables the P2P node to retrieve information about possible P2P neighbors such as the AS and some geographic information. This information can then be used by the P2P node to bias its neighbor selection. But purely using the "pWhois" service only helps the P2P system. It does not enable the ISP to express its preference and therefore does not enable cooperation.

Overview of Paper:
To overcome the argument that biasing the neighborhood selection process adversely affects the structural properties of the overlay topology one needs appropriate metrics. We propose metrics for evaluating the impact of using the oracle on the overlay as well as the underlay topology in Section 2 in addition to discussing how to derive realistic topologies.

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 service is not yet offered by the ISPs we are confined to using testlab facilities or simulators. Graph simulators enable us to explore large topologies as long as we focus purely on the graph properties. Packet level simulators enable us to incorporate the behavior of an actual P2P system but limit the complexity of the network that can be considered. Using testlab facility we can run the actual P2P system code and therefore no longer require to model it. Yet we again have to reduce the complexity of the network.

Accordingly we show in Section 3, relying on graph based simulations and measured Internet topologies, that the resulting P2P overlays maintain their graph properties like small diameter, small mean path length and node degree, but the densely connected subgraphs are now local to the ISPs. To study the impact of biased neighbor selection on a real P2P network that implements its own routing, we run extensive simulations of the Gnutella protocol in Section 4. These experiments help us to evaluate the effect of churn in P2P systems, and to study the impact of oracle on scalability and traffic content localization. We find that the Gnutella topologies maintain their graph properties, the ISP now has the ability to influence the overlay topology, and the scalability and network performance of Gnutella improves considerably. Then, in Section 5, we show that a modified version of Gnutella when used in a testbed can indeed take advantage of the oracle service. Finally, in Section 6, we summarize our findings and give an outlook on future work.

2. EVALUATION METHODOLOGY

In this section, we first propose metrics for evaluating the effectiveness of the idea of using an oracle which can also be used to characterize overlay-underlay graphs in general. Then we describe how we derive representative topologies for our simulations from the Internet AS topology.

2.1 Metrics

As a basic model for our investigations, we model the AS-graph as a complete bi-directed graph $G = (V,E)$ with a cost function $c : E \to \mathbb{R}^+$ associated with the edges. Every node represents an AS, and for every pair (u,v), let $c(u,v)$ denote the overall cost of routing a message from AS u to AS v (which depends on the routing policies of the ASes such a message may traverse).

Given a set of peers[1] P, let $AS : P \to V$ define how the peers are mapped to the ASes and $b : P \to \mathbb{R}^+$ denotes the bandwidth of the Internet connections of the peers. The overlay network formed by the peers is given as a directed graph $H = (P,F)$ in which every edge $(p,q) \in F$ has a cost of $c(AS(p),AS(q))$. The graph H can be characterized using several metrics.

2.1.1 Degree

The *degree* of a peer is defined as the number of its outgoing connections. Ideally, every peer should have a large number of connections to other peers within its AS so as to favor communication within the AS, while connections to other ASes should be limited to avoid high communication costs and high update costs as peers enter/leave the network.

2.1.2 Hop count diameter

Another parameter that should be small is the hop count diameter of the overlay graph H. The hop count diameter D of H is the maximum over all pairs $p,q \in P$ of the minimum length of a path (in terms of number of edges) from p to q in H. It is well-known that any graph of n nodes and degree d has a hop count diameter of at least $\log_{d-1} n$, and that dynamic overlay networks such as variants of the de Bruijn graph [26] can get very close to this lower bound, a very nice property. However, even though the hop count diameter may be small, the AS diameter (i.e., the distance between two P2P nodes when taking the underlying AS-graph G with cost function c into account) can be very large.

2.1.3 AS diameter

The AS diameter of H is defined as the maximum over all pairs $p,q \in P$ of the minimum cost of a path from p to q in P, where the cost of a path is defined as the sum of the cost of its edges. Ideally, we would like both the hop count diameter and the AS diameter to be as small as possible. Research in this direction was pioneered by Plaxton et al. [27], and the (theoretically) best construction today is the LAND overlay network [28].

Surprisingly, the best AS diameter achievable when avoiding many P2P connections to other ASes can be better than the best AS diameter achievable when all P2P connections go to other ASes. Consider the simple scenario in which the cost of a P2P edge within the same AS is 0 and that between two different ASes is 1. Let the maximum degree of a peer be d. In scenario 1, we require all edges of a peer to leave its AS, and in scenario 2, we only allow one edge of a peer to leave its AS. In scenario 1, the best possible AS diameter is $\log_{d-1} n$ (see our comments above). However, in scenario 2 one can achieve an AS diameter of just $\log_{d-2}(n/(d-1))$. For this, organize the peers into cliques of size $d-1$ within the ASes (we assume that the number of peers in each AS is a multiple of $d-1$). We can then view each clique as a node of degree $d-1$. It is possible to connect these nodes with a graph of diameter close to $\log_{d-2}(n/(d-1))$, giving the result above.

2.1.4 Flow conductance

Having a small hop count diameter and AS diameter is not enough to ensure high network performance. A tree, for example, can have very low hop count and AS diameter. Yet, it is certainly not a good P2P network, since one single faulty peer is sufficient to cut the network in half. Ideally, we would like to have a network that is well-connected so that it can withstand many faults and can route traffic with low congestion. A standard measure for this has been the expansion of a network. However, it seems that the expansion of a network cannot be approximated well. The best known algorithm can only guarantee an approximation ratio of $O(\sqrt{\log n})$ [29]. Therefore, we propose an alternative measure here that we call the *flow conductance* of a network (which is related to the flow number proposed in [30]).

Consider a directed network $G = (V,E)$ with edge bandwidths $b : E \to \mathbb{R}^+$. If $E(v)$ is the set of edges leaving v then for every node $v \in V$, let $b(v) = \sum_{e \in E(v)} b(e)$. Furthermore, for any subset $U \subseteq V$ let $b(U) = \sum_{v \in U} b(v)$. Next we consider the concurrent multicommodity flow problem M_0 with demands $d_{v,w} = b(v) \cdot b(w)/b(V)$ for every pair v,w of nodes. That is, we consider the heavy-traffic scenario in which each node aims at injecting a flow into the system

[1] In this paper a peer refers to a node of the P2P network and not to a BGP peer.

that is equal to its edge bandwidth, and the destinations of the flows are weighted according to their bandwidth. The *flow conductance* C measures how well the network can handle this scenario, or more formally, the flow conductance is equal to the inverse of the largest value of λ so that there is a feasible multicommodity flow solution for the demands $\lambda d_{v,w}$ in G. It is easy to show that for any network G, $0 \leq \lambda \leq 1$, and the larger λ is, the better is the network. As an example, for uniform link bandwidths the flow conductance of the $n \times n$-mesh is $\Theta(1/n)$ and the flow conductance of the hypercube of dimension n is $\Theta(1/\log n)$.

Interestingly, one can significantly lower the number of inter-AS edges without losing much on the flow conductance. Suppose we have m peers with bandwidth b that can have a maximum degree of d. Consider a class of networks $G(n)$ of degree d and size n with monotonically increasing flow conductance $C(n)$. Connecting the m peers by $G(m)$ gives a network with flow conductance $C(m)$. Suppose now that every peer can establish only one inter-AS edge with bandwidth $b/2$, and the remaining bandwidth can be used for intra-AS edges. In this case, let us organize the peers into cliques of size $d-1$ within the ASes (we assumed that the number of peers in each AS is a multiple of $d-1$) and interconnect the cliques so that they form $G(m/(d-1))$. Then it is not difficult to see that the resulting network has a flow conductance of $2C(m/(d-1))$. Hence, compared to arbitrary networks we lose a factor of at most two.

Summary:
We propose measures that are useful for P2P systems and our results demonstrate that it is possible to have a highly local topology with an AS diameter and a flow conductance that is comparable to the best non-local topologies. Hence, worst-case communication scenarios can be handled by local topologies (i.e., topologies with many intra-AS connections) essentially as well as by non-local topologies. In addition, we expect local topologies to be far better cost-wise for serving P2P traffic in practice than non-local topologies, which we aim to validate through experiments.

2.2 Simulation Topologies

The simulation results can be heavily influenced by the topologies used. Hence, we make the basis for our simulations the current AS topology of the Internet [31, 32], as it can be derived from the BGP routing information. We use BGP data from more than 1,300 BGP observation points including those provided by RIPE NCC, Routeviews, GEANT, and Abilene. This includes data from more than 700 ASes as on November 13, 2005. Our dataset contains routes with 4,730,222 different AS-paths between 3,271,351 different AS-pairs. We derive an AS-level topology from the AS-paths. If two ASes are next to each other on a path, we assume that they have an agreement to exchange data and are therefore neighbors. We are able to identify 58,903 such edges. We identify level-1 providers by starting with a small list of providers that are known to be level-1. An AS is added to the list of level-1 providers if the resulting AS-subgraph between level-1 providers is complete, that is, we derive the AS-subgraph to be the largest clique of ASes including our seed ASes. This results in the following 10 ASes being referred to as level-1 providers: 174, 209, 701, 1239, 2914, 3356, 3549, 3561, 5511, 7018. While this list may not be complete, all found ASes are well-known level-1 providers. There are 7,994 ASes that are neighbors of a `level-1` provider, which we refer to as `level-2`. All other 13,174 ASes are grouped together into the class `level-3`. We thus identify 21,178 ASes in all.

As it is not known how many P2P nodes are in each AS, and we may want to study smaller subsets to be able to compute the complex graph properties in reasonable time, we randomly subsample the AS-topology by keeping all level-1 ASes and their interconnections, and selecting a fraction of the level-2 and level-3 ASes while keeping their proportion the same as in the original data. Hereby, we first select the level-2 ASes and keep their interconnections. Only then do we select the level-3 ASes from among the ASes that are reachable in our subgraph.

Most level-1 ASes traditionally are expected to serve more customers than level-2 and level-3 ASes [33, 34]. At the same time there are more level-3 than level-2 than level-1 ASes. Thus we distribute the P2P clients among the ASes in the following ad-hoc manner: a P2P node has equal probability to pick an AS from each level. This results in a $1/3 : 1/3 : 1/3$ split of the nodes among the AS levels. This way a level-1 AS serves many more P2P nodes than a level-3 AS. All the topologies used in our experiments have been derived in this manner by randomly subsampling the AS topology derived from the BGP table dumps. Indeed, sensitivity analysis of our results show that if we move more peers to level-2, level-3 ASes the results improve even more.

3. OVERLAY / UNDERLAY GRAPH PROPERTIES

In this section, we first evaluate how the use of the oracle changes the graph properties of the P2P overlay topology. Later, in Sections 4 and 5 we explore the interactions of the two routing systems, the impact of churn on the topology, and the benefits of the oracle for satisfying queries. For this purpose we in this section use a general graph simulator as it allows us to explore large topologies. Namely, we rely on a simulation environment, the Subjects environment [35], that is very light-weight, such that we can run experiments on large topologies with many P2P nodes.

The Subjects environment is developed for the design of highly robust distributed systems and provides us with support for operations on general overlay graphs. It is based on C++ and consists of three basic types of entities: subjects, objects, and relay points. Subjects are the base class for processes (that are used to emulate nodes in the overlay network), objects are the base class for messages exchanged between subjects, and relay points are used by the subjects in order to establish connections to each other so that objects can be exchanged. In our experiments, the Internet class spawns multiple AS classes, each of which then spawns a number of overlay node classes. These nodes then establish peering connections with each other by exchanging messages (objects), and the relay points serve as an abstraction of network ports. The way these entities are set up ensures that subjects have a firm control on who can send information to them so that the consent and control principle can be strictly enforced.

For our evaluation we consider five graphs, each with 300 ASes and 4,372 P2P nodes, which results in an average of 14.6 nodes per AS. Each graph consists of 4 level-1 ASes, 100 level-2 ASes and 196 level-3 ASes. We place 375 nodes within each level-1 AS, 15 nodes within each level-2 AS, and 7 nodes within each level-3 AS. Increasing the number of nodes in the level-2, level-3 ASes only helps our case.

We establish P2P neighbor relationships by randomly picking one of the P2P nodes and let it establish a neighborship either

unbiased: to a single randomly chosen P2P node or
biased: to one from a list of candidates.

The unbiased case corresponds to a P2P protocol with arbitrary neighbor selection, while the biased case corresponds to a P2P node giving a list of potential neighbors to the oracle, and the oracle

helping it pick an optimal neighbor. We simulate the simplest of such oracles where it either chooses a neighbor within the querying node's AS if such a one is available, or a node from the nearest AS (considering AS hop distance). We experiment with different sizes of the oracle's choice list.

We experimented with establishing from 1000 upto 40,000 neighbor relationships in total. Given that for random graphs, the threshold for the number of edges to ensure connectivity is $\log n/2$ times the number n of nodes, it is not surprising that we need roughly 18,000 edges to ensure that the simulated graph is connected. Increasing the number of edges beyond this number does not change the graph properties noticeably. Accordingly, we concentrate on results for 20,000 peerings.

We run four experiments for each of the five AS graphs where the oracle is used for each neighbor relationship with candidate lists of length 1, 10, 50, 100, 200, 375, resulting in 120 experiments. Note that a list length of 1 corresponds to the unbiased case. The results we obtained are as follows.

Structural properties:
First, we check whether the overlay graphs remain connected using biased neighbor selection. In principle it is possible that due to a heavy bias, the graph disintegrates into disconnected components which are themselves well connected. We experimentally verify that all resulting graphs remain connected, thereby not impacting the reachability of the overlay graph.

The next question is if the mean degree of the P2P nodes changes. We find that the mean degree value of 9.138 of an unbiased graph changes to 8.8 in biased graphs with list size 200, see Figure 1(a). The small change in node degree implies that we do not affect the structural properties of the overlay graph seriously.

One may expect that our biased neighborhood selection increases the diameter and mean path length, as it prefers "closeby" neighbors. Yet, in all experiments the hop count diameter of the overlay graph stays at 7 or 8 hops and the AS diameter of the underlying AS graph stays at 5 hops. Neither does the average path length in the overlay graph increase significantly, see Figure 1(b). Therefore we can conclude that the biased neighborhood selection does not negatively impact the structural properties of the overlay graph.

Locality in topology:
We find that locality in overlays improves significantly as captured by the average AS-distance of P2P neighbors. Figure 1(c) shows how the AS-distance improves with the ability of the P2P node to choose a nearby neighbor. A lower AS-distance should correspond to lower latency. This is also reflected in the number of P2P neighbor connections that stay within each of the ASes, see Figure 1(d). Without consulting the oracle, only 4% of the edges are local to any of the ASes. The use of the oracle increases locality by a factor of 7 from 697 to 5088 (in a total of 20,000 peerings), even with a rather short candidate list of length 10. With a candidate list of length 200, more than half of the edges, 59%, stay within the AS. We find that the effects are even more pronounced for smaller networks. This demonstrates how much the oracle increases the ability of the AS to keep traffic within its network and with a refined oracle to better manage the P2P traffic. These results also indicate the benefit to the user, as traffic within the AS is less likely to encounter network bottlenecks than inter-AS traffic.

Flow conductance:
The remaining question is if the network maintains its ability to route traffic with low congestion. Since the run time requirements of our algorithm for computing a lower bound for the flow conductance of a graph is $O(n^4)$, we can currently only estimate the flow conductance for small graphs[2]. Being able to calculate the conductance of smaller graphs only is not a big problem as in case of Gnutella [15], we can calculate the conductance of the graph of ultrapeers, which is naturally much smaller than the entire Gnutella connectivity graph. We construct unbiased as well as biased graphs with 10 nodes and 21 edges, respectively 18 nodes and 51 edges. Both graphs are generated on a topology with 6 ASes.

The expected flow conductance of the unbiased graphs is 0.505 for the 10 node graph and 0.533 for the 18 node graph (see Section 2). We experimentally verify that both unbiased graphs support a conductance of at least 0.5. Also, we find that the penalty for the two biased graphs is less than a factor of 2. The 10 node biased graph supports a flow conductance of at least 0.3, and the 18 node graph, of at least 0.25. We furthermore observe that subgraphs of the biased graphs support a higher flow conductance which indicates that the connectivity within the ASes is good. This will likely result in a performance boost if the desired content can be located within the proximity of the interested user. The locality of biased graphs increases to 50% (for 10 nodes), respectively 80% (for 18 nodes) compared to 20% in the unbiased graphs.

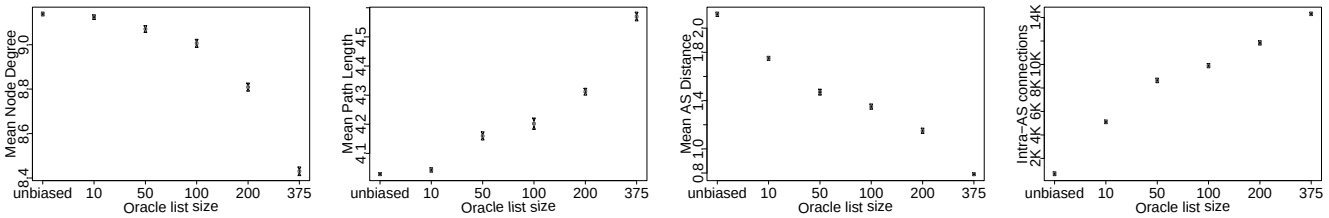
4. SIMULATIONS WITH AN ACTUAL P2P SYSTEM

In the previous section, we have seen that the results of biased neighbor selection on the graph properties of a generalized overlay network as well as its correlation to the underlay graph are promising. We now explore how a real P2P file sharing system benefits from using the oracle using a packet level network simulator [36]. For this purpose, we choose Gnutella, an unstructured P2P file sharing system. In the following we first give an overview of the Gnutella protocol, then discuss how we realize it within the simulation framework, and then discuss the simulation setup and our results.

4.1 Gnutella and SSFNet

Gnutella [15] is a popular file-sharing network with about 2 million users [12, 37]. Moreover it is an open-source system, which has attracted a healthy interest from researchers, e.g., [37, 38, 39]. The Gnutella network is comprised of agents called servents, who can initiate as well as serve requests for resources. When launched, a servent searches for other peers to connect to by sending Hello-like `Ping` messages. The `Ping` messages are answered by `Pong` messages, which contain address and shared resource information. The search queries are flooded within the Gnutella network using `Query` messages, and answered by `QueryHit` messages. To limit flooding Gnutella uses TTL (time to live) and message IDs. Each answer message (`QueryHit`/`Pong`) traverses the reverse path of the corresponding trigger message. While the negotiation traffic is carried within the set of connected Gnutella nodes, the actual data exchange of resources takes place outside the Gnutella network, using the HTTP protocol. Due to scalability problems, new versions of Gnutella take advantage of a hierarchical design in which some servents are elevated to ultrapeers, while others become leaf nodes. Each leaf node connects to a small number of ultrapeers, while each ultrapeer maintains a large number of neighbors, both ultrapeers and leafs. To further improve performance and to discourage abuse, the `Ping`/`Pong` protocol underwent semantic changes. Answers to `Ping`s are cached (Pong caching) and too frequent `Ping`s or repeated `Query`s may cause termination of connection.

[2]Meanwhile, we have found a way to reduce the complexity to $O(n^2 logn)$ and work on computing the conductance of larger graphs is continuing.



(a) P2P node degree (b) Overlay path length (c) Underlay AS distance (d) Intra-AS P2P connections

Figure 1: Error plots showing comparison of metrics with increasing size of Oracle list.

We have coded the Gnutella protocol within the packet level network simulator SSFNet [40]. The Scalable Simulation Framework (SSF) [36] is an open standard for simulating large and complex networks. Written in Java, it supports discrete-event simulations. 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, BGP and OSPF, network elements like hosts, routers, links, and LANs, and their various support classes. The network topologies are defined using the Domain Modeling Language (DML), and the SSFNet class instances autonomously configure and instantiate themselves by querying these DML configuration files. The coding for the lower layers of the IP stack is thus provided by SSFNet, while we implement the Gnutella protocol as an SSFNet application [40].

We modify the neighbor selection procedure of Gnutella to take advantage of the oracle [41]. Normally, when a Gnutella node connects to the network, it gets a list of popular Gnutella node addresses in its Hostcache [42], which is a locally maintained Gnutella hosts list, typically containing a few hundred IP addresses. The node chooses a random subset of the Hostcache, and initiates Gnutella peerings with these selected nodes. We modify this procedure so that the Gnutella node sends the contents of its Hostcache (list of IP addresses) to the oracle, which then 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 neighborhood selection of Gnutella network, to choose a peer within the AS if it exists. Moreover when a Gnutella node receives query results for its search requests, it again consults the oracle to select the nearest node from whom it then downloads the file content.

4.2 Simulation setup

The topologies are derived using the methodology explained in Section 2.2. The network consists of a total of 25 ASes and 1000 nodes. More specifically it consists of 1 level-1 AS, 8 level-2 ASes and 16 level-3 ASes. We place 360 nodes within the level-1 AS, 40 nodes within each level-2 AS, and 20 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 25 ASes, 50 routers and 1000 nodes running the Gnutella protocol.

Each leaf node can have between 2 to 4 connections to ultrapeers, while each ultrapeer initiates at least 10 connections to other Gnutella nodes itself, and stops accepting incoming connections from other nodes, once it is connected to 45 nodes, be they leafs or ultrapeers. Each node shares between 0 and 100 files, uniformly distributed.

To take churn in P2P systems into account, each node remains online for a minimum of 1 and a maximum of 1500 seconds. Once a node goes off-line, it may become online again after a time period between 1 to 300 seconds. For a start, we take these time periods as uniformly distributed but we are in the process of migrating to more precise distributions, as recently revealed in [39]. Furthermore, a leaf node must be online for at least 600 seconds before it can serve as an ultrapeer. At any given point of time in our simulations, we find that $20-25\%$ nodes are off-line[3], and a quarter of the online nodes are functioning as ultrapeers.

We ran multiple simulations for arbitrary lengths of time and found that the startup phase of the simulation lasts for about 500 seconds. After 5000 seconds of simulation time, the summary statistics do not show significant changes. Therefore we run our simulations for 5000 seconds.

4.3 Results

We first analyze the Gnutella network graph according to the metrics explained in Section 2, followed by an evaluation of some Gnutella specific metrics like scalability of network, number of messages exchanged, localization of file content exchange and visualization of topology.

We run three different experiments on five different topology instances with roughly the same number of search queries and the following parameters for the Gnutella nodes:

- Cache size = 1000, without oracle
- Cache size = 100, with oracle for neighbor selection
- Cache size = 1000, with oracle for neighbor selection

Note that in our implementation, each Gnutella node sends the contents of its Hostcache to the oracle, which ranks the list of IP addresses according to proximity from the querying node. In other words, the above three cases correspond to experiments with oracle list size of 1, 100, and 1000 respectively. The success rates of the search queries are similar.

To explore the influence of consulting the oracle on the network topology we visualize, in Figure 2 [41], the Gnutella overlay topology, for the unbiased case and the biased case with oracle list size 1000. At a particular instant in time, we sample the Gnutella overlay topology, display all the online nodes in the graph, and join two nodes with an edge if there exists a Gnutella peering between them at this point of time. Then, using the visualization library yWorks [44], we convert both the graphs into a structured hierarchical format. The resulting graph structures are displayed in Figure 2.

[3]This is more aggressive as compared to other studies, e.g., [43] which assume that only half the nodes churn.

We can easily observe that the Gnutella topology in the biased case is well correlated with the Internet AS topology, where the nodes within an AS form a dense cluster, with only a few connections going to nodes in other ASes. This is in stark contrast to the unbiased Gnutella graph, where no such property can be observed.

To analyze how churn influences the metrics such as node degree, path length, diameter and number of intra-AS peerings, we sample the Gnutella network 10 times during the simulation run, i.e., every 500 seconds. The results are shown in Figure 3. Multiple runs of the above experiments, using different world topologies yield similar results.

Graph connectivity:
We begin by checking whether the Gnutella network graph remains connected using biased neighbor selection. We define the Gnutella network graph at a particular time instant as the graph formed by nodes that are online at that instant, where two nodes are connected by an edge if there exists a Gnutella connection between them at that instant. We experimentally verify that the Gnutella network remains connected at all 10 times where we sample the network, for all three cases. Hence, biased neighbor selection does not affect the connectivity of Gnutella network.

Mean Node Degree:
Since ultrapeers have a much larger node degree than leaf nodes, we show, in Figure 3(a) and (b), how the mean node degree changes over time in a barplot for all three cases separately for ultrapeers and leaf nodes. This enables us to check if a biased neighbor selection affects the structural properties of Gnutella adversely. We observe that the mean node degree for leafs decreases only slightly, across time, with a maximum decrease from 3.14 to 2.08 at 3500 seconds. The same is the case for ultrapeers, where the maximum decrease is from 15.29 to 10.75, again at 3500 seconds. In other words, despite biasing the neighbor selection via the oracle, the node degree for both leafs and ultrapeers stays within the expected range, and the network structure of Gnutella remains unchanged.

Graph diameter:
The diameter of the overlay graph, which is $5-7$ hops in the unbiased case, increases to $6-8$ hops with a oracle size of 100, only a nominal increase. Using an oracle with list size of 1000 results in a diameter between $7-12$ hops, with an average of 9.2. The AS diameter of the underlay graph remains is 4 hops in all cases.

Mean Overlay path length:
The average path length in the Gnutella overlay, shown in Figure 3(c), while registering an increase, does not change significantly. The maximum increase occurs at 3500 seconds, from 3.35 in the unbiased case to 5.21 hops in the biased case with oracle list size of 1000.

Mean AS distance:
The benefits of using an oracle for biasing the neighborhood in Gnutella are visible in Figure 3(d), which shows the average AS distance (in the underlay) between any two connected Gnutella nodes. The AS distance is obtained as follows. We map each Gnutella node's IP address to its parent AS, and for each overlay edge, we find the network distance in AS hops between the two end-nodes. We observe that the least amount of decrease in the average AS distance occurs from 1.93 to 0.8 at 1000 seconds, and the maximum decrease from 1.94 to 0.25 happens at 5000 seconds. Given that the AS diameter remains constant at 4 hops, the average decrease of 1.45 in the AS distance is significant. Besides, as the average AS distance in the case of oracle list size of 1000 is 0.45, a value less than 1, it implies that most of the Gnutella peerings are indeed within the ASes, i.e., they are not crossing AS boundaries. This can be a major relief for ISPs, as they do not incur any additional cost for traffic within their domains. Also traffic that does not leave the network is easier to manage. Moreover, P2P traffic will not encounter inter-ISP bottlenecks.

Intra-AS P2P connections:
The above observations on AS distance are even better understood from the plots in Figure 3(e) and (f), where we show the total number of intra-AS P2P connections in the Gnutella network as a percentage of the total number of intra- and inter-AS P2P connections, for both leafs and ultrapeers.

In Figure 3(e), we observe that in the case of leaf nodes, taking the average over the 10 time points, the percentage of intra-AS P2P connections increases from 14.6% in unbiased case to 47.88% in the case of oracle with list size 100. For oracle with list size 1000, we note an average of 82.22% intra-AS P2P connections.

In Figure 3(f), we observe similar results for ultrapeers. The percentage of intra-AS P2P connections increases from an average value of 14.54% in the unbiased case to 38.04% in the case of oracle with list size 100, and further to 74.95% in case of oracle with list size 1000.

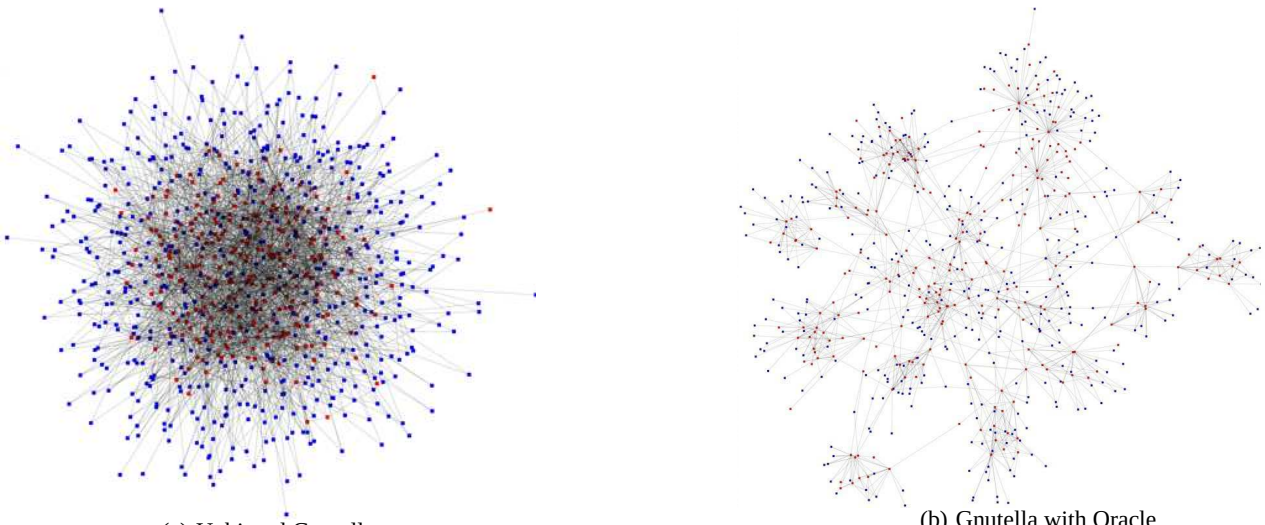
The percentage increase in intra-AS P2P connections is larger for leaf nodes as compared to ultrapeers, a welcome development. One needs a certain number of inter-AS connections, to maintain network connectivity and to be able to search for file content that may not be available within an AS. However, as leaf nodes typically have poor connectivity to the Internet, and have lower uptimes, it is reasonable to have leaf nodes keep most of their peerings within their AS, while allowing the ultrapeers to have slightly more connections outside their ASes.

Overall, we observe that the results for the metrics comparison in Gnutella simulations are in conformity with the graph-based simulation results in Section 3.

Scalability of Gnutella:
In order to quantify the impact of biased neighborhood selection on the scalability of the Gnutella network, we measure the number of Gnutella messages generated in the entire network, for all the three cases. The negotiation traffic in many P2P systems like Gnutella represents a large portion of the total P2P traffic [38]. In Table 1, we show the number of each type of Gnutella message (`Ping`, `Pong`, `Query` and `QueryHit`) generated during the entire simulation run. Note that the number of unique messages generated is about the same in all the three cases. However, when a `Ping` or `Query` is generated by a node, and flooded to its `n` neighbors, the message is counted `n` times. Hence, the table shows the total number of messages exchanged between Gnutella nodes.

As we can observe, the number of `Ping` messages decreases from 7.6 million in the unbiased case to 4 million in the case of oracle with list size 1000. Even more significant is the reduction of `Pong` messages, from 75.5 million to 39 million messages. The `Query` and `QueryHit` messages also register similar improvements. This reduction of `Ping/Pong` messages by a factor of 2, and search queries by a factor of almost 3 proves that the scalability of Gnutella network improves significantly with biased neighborhood selection.

The reason for this reduction in message volume is as follows. Even though the node degrees are largely unchanged, the oracle helps in building an efficient overlay topology. As the nodes form a dense cluster within an AS with very few inter-AS connections, caching of messages ensures that messages are flooded within sub-



(a) Unbiased Gnutella

(b) Gnutella with Oracle

Figure 2: Visualization of Gnutella overlay topology

Gnutella Message Type	Unbiased Gnutella	Biased, cache 100	Biased, cache 1000
Ping	7.6M	6.1M	4.0M
Pong	75.5M	59.0M	39.1M
Query	6.3M	4.0M	2.3M
QueryHit	3.5M	2.9M	1.9M

Table 1: Number of exchanged Gnutella message types

networks very efficiently, by traversing lesser overlay hops, which is reflected in Table 1. Thus information is propagated with lesser message hops, lower delays and reduced network overhead.

Localization of content exchange:
The negotiation traffic traverses within the set of connected Gnutella nodes, but the actual file content exchange happens outside the Gnutella network, using the standard HTTP protocol. When a Gnutella node gets multiple `QueryHits` for its search query, it chooses a node randomly and initiates an HTTP session with it to download the desired file content. Since the file content is often bulky, it is prudent to localize this traffic as well, as it relates directly to user experience. In the above experiments, we use the oracle to bias only the neighborhood selection. In other words, when a node comes online, it consults the oracle and sends connection requests to an oracle-recommended node selected from its Hostcache. However, while choosing a node from the `QueryHits`, it so far did not consult the oracle. We now analyse how much of the file content exchange remains local in this case and how much one can gain if one consults the oracle again at this stage.

We observe that the intra-AS file exchange, which is 6.5% in the unbiased case, improves slightly to 7.3% in case of oracle with list size 100, and to 10.02% in case of oracle with list size 1000.

We then further modify the neighborhood selection, so that a node consults the oracle again at the file-exchange stage, with the list of nodes from whom it gets the `QueryHits`. After this change, we notice that 40.57% of the file transfers now occur within an AS. In other words, 34% of file content, which is otherwise available at a node within the querying node's AS, was previously downloaded from a node outside the querying node's AS.

This leads us to conclude that consulting the oracle for neighborhood selection, during bootstrapping stage as well as file-exchange stage, leads to significant increase in localization of P2P traffic.

5. TESTLAB EXPERIMENTS

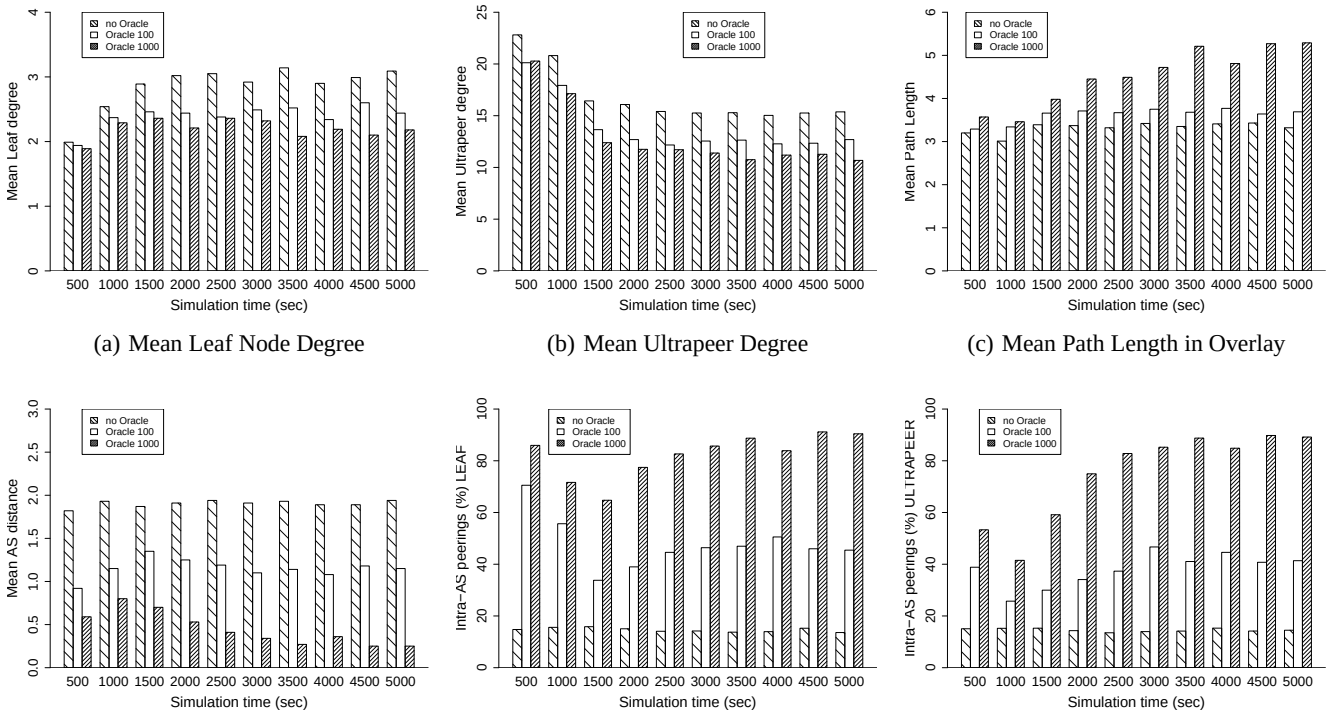
After extensive simulations on general overlay graphs and Gnutella system, we now confirm these results by modifying P2P clients, namely Gnutella, to take advantage of the oracle service in a controled setting, a Testlab.

Using 5 routers, 6 switches, and 15 computers, we configure four different 5-AS topologies: ring, star, tree and random mesh. Each router is connected to 3 machines, and each machine runs 3 instances of Gnutella software, where one is an ultrapeer and the other two are leaf nodes. Thus, we have a network of 45 Gnutella nodes, each running the GTK-Gnutella software [45]. A router is taken as an abstraction of an AS boundary.

We modify the source code of the Gnutella nodes, so that when a node wishes to join the network, it sends the contents of its Hostcache to the oracle. The Hostcache of each node is filled with a random subset of the network nodes' IP addresses. The oracle is a central machine accessible to all Gnutella nodes, and running the oracle's neighbor selection algorithm. When it gets a list of IP addresses from a node, it ranks the list according to AS hops distance. Hence, the Gnutella node joins another node within its AS if such a node is present in its Hostcache, else it joins a node from the nearest AS.

We experiment with two schemes of file distribution. In the uniform scheme, each node shares 6 files each. In the variable scheme, each ultrapeer shares 12 files, half the leaf nodes shares 6 files each, and the remaining leaf nodes share no content. We thus have 270 unique files with real content.

We run two sets of experiments: unbiased Gnutella and Gnutella using oracle. We generate 45 unique search strings, one for each node, and allow each node to flood its search query in the network. Each node searches for the same query string in both the experiments. We then calculate the total number of `Query` and `QueryHit` messages exchanged in the network and analyze whether biased neighbor selection leads to any unsuccessful content search which was otherwise successful in unbiased Gnutella. We



(a) Mean Leaf Node Degree (b) Mean Ultrapeer Degree (c) Mean Path Length in Overlay

(d) Mean AS distance in Underlay (e) Intra-AS peerings (%) for Leaf nodes (f) Intra-AS peerings (%) for Ultrapeers

Figure 3: Plots showing comparison of metrics in Gnutella simulations

experimentally verify that all `Query`s that are satisfied in unbiased Gnutella network, are also satisfied in the biased Gnutella network. We find, as predicted by the simulations, that with biased neighbor selection, the number of `Query` and `QueryHit` messages decreases (60% reduction in `Query`, 12% reduction in `QueryHit`) and that the messages tend to stay within the ASes.

6. SUMMARY AND FUTURE WORK

P2P systems build their overlay topology largely agnostic of the Internet underlay. To overcome this, we propose to use an oracle hosted by the ISPs, so that ISPs and P2P users can cooperate for improved performance. Such an oracle can be queried by P2P nodes while choosing neighbors or while deciding from whom to download content and it will rank the possible neighbors of the querying node according to a locality indication. We propose metrics to evaluate the effectiveness of using an oracle and show that using the oracle allows the overlay topologies to maintain the graph properties like small diameter, small mean path lengths and node degree, while at the same time, tremendously increasing their network locality (lesser mean AS distance, larger number of intra-AS peerings). Even the ability of the network to route arbitrary traffic patterns with low congestion, while reduced, is still reasonable. These results along with results on improved scalability and network performance are obtained relying on graph based simulations, packet level simulation of an actual P2P system, as well as experiments with a modified P2P client in a testlab.

We are in the process of experimenting with the oracle scheme in Planetlab to increase the scale of our experiments and to test the interactions of the modified P2P clients with unmodified ones. We have realized the oracle as a Web server, which relies on a dynamic database and are in the process of installing Gnutella and Bittorrent clients on Planetlab nodes. The Bittorrent client will consult the oracle once it gets the node list from the tracker. Alternatively the tracker may consult the oracle, to keep its list of Bittorrent nodes sorted according to distance from the querying nodes.

As more of the P2P traffic is localized within an ISP the available bandwidth may increase as it is no longer capped by the peering links [10]. This could lead to a usage increase which in turn may again complicate the traffic engineering problem. Yet, even this situation can be addressed by the oracle, as it can take the ISP topology and its bottlenecks into account when trying to rank the possible P2P clients.

In a next step we plan to design simple, provably good, and experimentally well-behaved distributed algorithms for P2P neighborhood selection that take full advantage of such an oracle. We want to experiment with recent revelations of user behaviour and file sharing distributions (e.g. [38, 39]) in SSFNet, and also wish to compare the performance of oracle with latency-based protocols for neighbor selection. Computing the flow conductance of larger graphs, and exploring its relationship with lower resilience to churn is another task. An important issue that we intend to investigate is the trade-off between locality and congestion. Certainly, if strict locality is enforced (i.e., a file is always retrieved from the closest peer), there are situations where peers can encounter a high congestion. Hence, flexible schemes are needed that will fetch files from nearby peers if there is no congestion and otherwise will switch to more remote peers. This will eventually enable us to develop a theoretical model to investigate the question, what is the optimal level of locality for an overlay system.

Acknowledgements
We would like to thank Rumen Tashev for helping us with SSFNet simulations, and the anonymous reviewers for their valuable comments. Support for this work was partially provided by EU FP6 project DELIS.

7. REFERENCES

[1] A. Nakao, L. Peterson, and A. Bavier, "A Routing Underlay for Overlay Networks," in *SIGCOMM*, 2003.
[2] "Slyck," `http://www.slyck.com/`.
[3] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos, "Is P2P dying or just hiding?," in *GLOBECOM*, 2004.
[4] Light Reading, "Controlling P2P Traffic," `http://www.lightreading.com/document.asp?site=lightreading&doc_id=44435&page_number=3`.
[5] R. Steinmetz and K. Wehrle, *P2P Systems and Applications*, Springer Lecture Notes in CS, 2005.
[6] T. Mennecke, "DSL Broadband Providers Perform Balancing Act," `http://www.slyck.com/news.php?story=973`.
[7] R. Keralapura, N. Taft, C. Chuah, and G. Iannaccone, "Can ISPs Take the Heat from Overlay Networks?," in *HotNets*, 2004.
[8] G. Shen, Y. Wang, Y. Xiong, B. Zhao, and Z. Zhang, "HPTP: Relieving the Tension between ISPs and P2P," in *IPTPS*, 2007.
[9] V. Aggarwal, S. Bender, A. Feldmann, and A. Wichmann, "Methodology for Estimating Network Distances of Gnutella Neighbors," in *GI Jahrestagung - Informatik 2004*, 2004.
[10] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should ISPs fear Peer-Assisted Content Distribution?," in *IMC*, 2005.
[11] A. Akella, S. Seshan, and A. Shaikh, "An Empirical Evaluation of Wide-Area Internet Bottlenecks," in *ACM IMC*, 2003.
[12] A. Rasti, D. Stutzbach, and R. Rejaie, "On the Long-term Evolution of the Two-Tier Gnutella Overlay," in *Global Internet*, 2006.
[13] S. Halabi, *Internet Routing Architectures*, Cisco Press, 2000.
[14] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *SOSP*, 2001.
[15] "Gnutella v0.6 RFC," `http://www.the-gdf.org/`.
[16] S. Savage, A. Collins, and E. Hoffman, "The End-to-End Effects of Internet Path Selection," in *SIGCOMM*, 1999.
[17] S. Seetharaman and M. Ammar, "On the Interaction between Dynamic Routing in the Native and Overlay Layers," in *INFOCOM*, 2006.
[18] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically aware overlay construction and server selection," in *INFOCOM*, 2002.
[19] K. Shanahan and M. Freedman, "Locality Prediction for Oblivious Clients," in *IPTPS*, 2005.
[20] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, and D. Yao, "Optimal Selection of Peers for P2P Downloading and Streaming," in *INFOCOM*, 2005.
[21] Bindal et.al., "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *IEEE ICDCS*, 2006.
[22] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, "Brocade: Landmark Routing on Overlay Networks," in *IPTPS*, 2002.
[23] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *SIGCOMM*, 2002.
[24] "pWhoIs," `http://pwhois.org`.
[25] "Cymru Whois," `http://www.cymru.com/BGP/asnlookup.html`.
[26] M. Naor and U. Wieder, "Novel architectures for P2P applications: the continuous-discrete approach," in *SPAA*, 2003.
[27] G. Plaxton, R. Rajaraman, and A. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *SPAA*, 1997.
[28] I. Abraham, D. Malkhi, and O. Dobzinski, "LAND: stretch $(1+\varepsilon)$ locality-aware networks for DHTs," in *SODA*, 2004.
[29] S. Arora, S. Rao, and U. Vazirani, "Expander flows, geometric embeddings and graph partitioning," in *STOC*, 2004.
[30] P. Kolman and C. Scheideler, "Improved bounds for the unsplittable flow problem," in *SODA*, 2002.
[31] W. Muehlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig, "Building an AS-Topology Model that Captures Route Diversity," in *SIGCOMM*, 2006.
[32] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic Topology Analysis and Generation Using Degree Correlations," in *SIGCOMM*, 2006.
[33] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-level Topology," in *SIGCOMM*, 2004.
[34] H. Chang, S. Jamin, Z. Mao, and W. Willinger, "An Empirical Approach to Modeling Inter-AS Traffic Matrices," in *IMC*, 2005.
[35] C. Scheideler, "Towards a paradigm for robust distributed algorithms and data structures," in *HNI Symposium on New Trends in Parallel and Distributed Computing*, 2006.
[36] "SSFNet," `http://www.ssfnet.org`.
[37] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems," in *ACM IMC*, 2005.
[38] A. Gish, Y. Shavitt, and T. Tankel, "Geographical Statistics and Characteristics of P2P Query Strings," in *IPTPS*, 2007.
[39] D. Stutzbach and R. Rejaie, "Understanding Churn in P2P Networks," in *IMC*, 2006.
[40] V. Aggarwal, A. Feldmann, and S. Mohrs, "Implementation of a P2P system within a network simulation framework," in *ECCS P2P-Complex Workshop*, 2005.
[41] R. Tashev, "Experimenting with Neighbour Discovery Schemes for P2P Networks in a Simulation Framework," in *Master thesis, Dept of CS, TU Munich*, 2006.
[42] "Gnutella Hostcache," `http://www.the-gdf.org/index.php?title=The_Local_Hostcache`.
[43] P. Linga, I. Gupta, and K. Birman, "A Churn-Resistant P2P Web Caching System," in *SSRS*, 2003.
[44] "yWorks," `http://www.yworks.com/`.
[45] "GTK-Gnutella," `http://www.gtk-gnutella.com/`.