P4P: Provider Portal for (P2P) Applications

Haiyong Xie
Y. Richard Yang
Avi Silberschatz
Yale University

Arvind Krishnamurthy
University of Washington

Laird Popkin
Pando Networks, Inc

ABSTRACT

The emergence of peer-to-peer (P2P) is posing significant new challenges to achieving efficient and fair utilization of network resources. In particular, without the ability to explicitly communicate with network providers, P2P applications depend mainly on inefficient network inference and network-oblivious peering, leading to potential inefficiencies for both P2P applications and network providers. To address the issues, we propose a simple, light-weight framework called P4P to allow more effective cooperative traffic control between applications and network providers. Our evaluations show clear performance benefits of the framework. The formation of the P4P working group consisting of major ISPs and P2P developers will further develop the framework.

1. INTRODUCTION

A basic problem in a network architecture is how network applications (*i.e.*, network resource consumers) efficiently utilize the network resources owned by network providers. We refer to this problem as the network efficient traffic control problem, or traffic control for short. This problem is particularly important as it can have significant impacts on application performance, network efficiency and economics, and overall system complexity.

In the current Internet, for traditional point-to-point applications, efficient traffic control is largely determined by network providers alone: applications specify only the destinations of traffic; it is up to the network to control both the paths taken by the traffic and the transmission rates (through TCP feedback) on the chosen paths. Network providers can therefore improve efficiency unilaterally according to their objectives. Specifically, providers can use optimal traffic engineering to determine efficient routing and satisfy economical objectives such as implementing valley-free routing.

However, the recent emergence of P2P applications is posing significant challenges to efficient traffic control, with neither the network nor the P2P system having complete leverage over system efficiency.

First, for intradomain, the network-oblivious peering strategy of many P2P applications may cause traffic to scatter and unnecessarily traverse multiple links within a provider's network, leading to much higher load on some backbone links and performance degradation to other applications (see, *e.g.*, [8, 10, 18])). Second, for interdomain, network-oblivious peering may cause a P2P application in a non-tier-1 network provider to relay a substantial amount of traffic between its providers [13]. This may lead to serious disruption of ISP economics (see, *e.g.*, [4, 16]). Even for tier-1 ISPs who do not make payments to network providers, P2P traffic may cause traffic imbalance between its peers, leading to potential violation of peering agreements. Third, P2P's dynamic traffic distribution patterns do not necessarily enjoy a synergistic coexistence with network traffic engineering [7, 12] – network providers go to great lengths to estimate traffic matrices and determine routing based on them, but all of this effort could be negated if P2P applications modify their download behavior to adapt to changes in the network, thereby resulting in oscillations in traffic matrices and sub-optimal routing decisions.

In response to these kinds of issues, network providers have considered multiple new traffic control techniques. Unfortunately, none of them appear to be fully satisfactory – without P2P cooperation, the new techniques are either ineffective or degrade P2P performance and often times are too complex. One approach, for example, is to install P2P caching devices to cut down bandwidth consumed by P2P applications (*e.g.*, [5, 6, 14, 15]). However, deploying these caches can be expensive for large networks. To be effective, the caches need to recognize P2P applications, limiting their generality and applicability to new or encrypted protocols. Furthermore, there are complex legal issues involved in caching. Another technique is to deploy traffic shaping devices to rate limit P2P (*e.g.*, [1, 2]). These devices rely on deep packet inspection or other P2P traffic identification schemes. However, different P2P protocols use different control messages, and many P2P protocols use encryption and dynamic ports to avoid being identified. It remains unclear whether in the long run traffic shaping can effectively control the bandwidth consumption of P2P applications and reduce provider's operational costs. Furthermore, unilateral rate limiting by network providers may substantially degrade P2P performance and be at odds with consumer's needs.

With network provider solution being ineffective, there are evidences that P2P applications trying to utilize peering flexibility to improve network efficiency. For example, several popular P2P applications such as Joost and Kontiki have tried to localize peering using the autonomous systems of peers. However, there are fundamental limits on what P2P can achieve alone: to improve efficiency, P2P applications will have to rely on inferring various types of network information such as topology, congestion status, cost, and in particular ISP policies. However, reverse engineering of such information is challenging if not impossible.

Overall, the P2P paradigm exposes a fundamental issue in traditional traffic control: emerging applications can have tremendous flexibility in how data is communicated, and thus, they should be an integral part of network efficient control. However, if end hosts are to participate in network resource optimizations, then the networks cannot continue to be opaque but need to provide a communication channel for collaborative traffic control.

In this position paper, we present a simple, flexible framework called P4P. Here P4P stands for provider portal for (P2P) applications. The P4P framework provides interfaces for applications to communicate with network resource providers regarding information such as resource provider capabilities, network info/policy, and virtual cost. The interfaces preserve provider privacy and allow network providers and P2P applications to jointly optimize their respective performance.

At the core of this framework is the *virtual cost* interface through which ISPs can communicate to P2P applications the current "peering costs" on its intradomain and interdomain links.[1] These costs reflect ISPs' preferences regarding P2P connectivity, and can be used to capture a number of interesting ISP metrics, such as peak backbone utilization, preferred interdomain links, and so on. The P2P systems use these costs to shape their connectivity and choose ISP-friendly communication patterns if possible. The interface provides a simple and clean decomposition between ISP and application operations. as a result, neither do the ISPs need to know the specifics of applications, nor do applications need to know the specifics of ISPs or other applications sharing network resources on the same ISP. In particular, the interface enables us to apply the primal-dual decomposition technique to derive virtual costs. Thereby, the principled interface design leads to extensibility, scalability and efficiency.

A Distributed Computing Industry Association (DCIA) working group, P4PWG, was formed in July 2007 to provide a forum for P2P providers, ISPs and researchers to work together to optimize efficiency and performance. The P4P Working Group has conducted a large-scale pilot study utilizing P4P. The working group consists of representatives from over 70 P2P providers, ISPs and researchers. Although there are many outstanding issues, our current evaluations show that P4P can improve performance for both P2P and ISPs. Thus, it is a framework that should be more thoroughly explored. In this position paper, we present the motivation and design perspective of P4P. There are other submissions to this workshop addressing other perspectives (e.g., implementation) of P4P.

2. THE P4P FRAMEWORK

The P4P framework is a flexible and light-weight framework that allows network providers to explicitly provide more information, guidelines and capabilities to emerging applications, such as P2P content distribution.

2.1 Motivation

We now motivate the need for a P4P portal to enable explicit communications between P2P and network providers.

First, P2P systems have tremendous flexibility in shaping their traffic flow. Given that a client interested in a piece of data can download it from any one of the multiple sites storing the data, there are clear benefits to be had in intelligently choosing a data source (or, alternately, choosing a peer in a tit-for-tat system). This flexibility fundamentally changes the traditional network traffic control problem, which is typically solved in the context of a given traffic demand matrix. In the updated setting, there are multiple ways of satisfying the data demands of an application, each resulting in a different traffic demand matrix, and an efficient solution would require the explicit involvement of the P2P application.

Second, the current network architecture allows only for limited, implicit communications between network providers and applications. In this setting, if a P2P application seeks to exploit the flexibility in controlling its data transfers to improve efficiency, it will have to probe the network to reverse engineer information such as topology, status and policies. However, this is rather challenging in spite of significant progress in network measurement techniques. For one thing, it is clearly redundant and wasteful to have each application perform probing. Even if this issue is addressed by a coordinated service for topology inference (*e.g.*, [9]) to reduce the overhead, the fundamental hurdle is the ability to perform the inference in an accurate manner. New technologies, such as MPLS, and routers that do not respond to measurement probes make it difficult to infer network characteristics. More importantly, available bandwidth and loss-rate estimation from end hosts are difficult because their views are obscured by last-mile bottlenecks; it is difficult for an end host to identify which links are under-utilized or over-utilized. Furthermore, cost and policy information are difficult, if not impossible, to reverse engineer. For example, it is difficult for P2P to determine which peers are accessible through lightly-loaded intradomain links and/or lower-cost interdomain links (where the cost takes into account factors such as inter-domain policies, traffic balance ratio between peering providers, and 95% percentile based billing).

In summary, for traditional applications, routing is made by network providers using a predictable traffic demand matrix with full network knowledge. With high levels of P2P traffic, the traffic control problem needs to be jointly solved by network providers and P2P applications.

2.2 Design Rationale

We consider the following design requirements.

- Better P2P performance. While some P2P systems exploit locality and network status to have its clients refine their peerings, the performance improvement is limited due to factors such as limited network information and slow convergence that is further exacerbated by churn [11]. Using more accurate network status information, P4P should be able to identify more efficient connections.
- More efficient network resource usage. By enabling explicit communication between P2P and the network, P4P can enable applications to use network status information to reduce backbone traffic and lower operation costs.
- Both application and ISP control. It should not be the

[1]These costs are not to be confused with payments made by ISP to its interdomain peers. Instead, they reflect the abstract costs that the ISP associates with application-level peering over these links.

case that one side dictates the behavior of the side. The design objective is to serve as communication channel for applications to make more informed decisions.

- Scalability. P4P should support a large number of users and P2P networks in very dynamic settings; any proposed information exchange and optimization techniques should be computationally inexpensive. It may not be scalable if each peer joining is handled by an ISP.
- Privacy preservation. P4P should address a major incentive concern of network providers who may want to preserve privacy when releasing their network information. Individual peers do not want to be tracked by ISPs.
- Extensibility. There are many types of P2P applications with varying features. For instance, P2P systems for file sharing and streaming might have different needs, such as P2P streaming having more stringent real-time constraints than file sharing. Also, some applications use trackers (referred to as *appTracker* hereafter) to bootstrap and guide peer selection, while others do not; in addition, peers may exchange information locally through gossip messages. P4P should be flexible to handle a wide range of P2P applications with varying requirements and features.
- Fault tolerance. Failure of P4P components should lead to only inefficiency, instead of system failure.
- Incremental deploymentability. We do not target a clean-slate re-design. The P4P framework should be incrementally deployable, one network provider at a time, one P2P application at a time.
- Provider contribution for P2P acceleration. A network provider may have many capabilities which it can provide to accelerate content distribution for P2P and at the same time increase its revenue. Examples include class of service, or quality of service that a P2P content provider can request. Also, a provider may contribute fixed servers as high-capacity seeds or caches, and this information should percolate to the P2P application.
- Open standard. Any ISP, provider, application can easily implement it.

2.3 Design Overview

The P4P framework consists of a data-plane component and a control-plane component.

In the data plane, P4P allows applications to mark importance of traffic. Also, routers on the data plane can give fine-grained feedback to P2P and enable more efficient usage of network resources. Specifically, routers can mark the ECN bits of TCP packets (or a field in a P2P header), or explicitly designate flow rates; end hosts then adjust their flow rates accordingly. For instance, a multihomed network can optimize financial cost and improve performance through virtual capacity computed based on 95-percentiles [3]. When the virtual capacity is approached, routers mark TCP packets and end hosts reduce their flow rates accordingly; thus the network provider can both optimize its cost and performance and allocate more bandwidth to P2P flows. We emphasize that the data plane component is optional and can be incrementally deployed.

In the control plane, P4P introduces *iTrackers* to provide portals for P2P to communicate with network providers. The introduction of *iTrackers* allows P4P to divide traffic control responsibilities between P2P and providers, and also makes P4P incrementally deployable and extensible.

Specifically, each network resource provider, be it a conventional commercial network provider (*e.g.*, AT&T), a university campus network, or a virtual service provider (*e.g.*, Akamai), maintains an *iTracker* for its network. A P2P client obtains the IP address of the *iTracker* of its local provider through DNS query (with a new DNS record type `P4P`) or another service such as WHOIS. Standard techniques can be applied to allow for multiple *iTrackers* in a given domain, especially for fault tolerance and scalability. An *iTracker* provides a portal for three kinds of information regarding the network provider: network capabilities; network status/topology; and provider guidelines/policies.

3. THE P4P CONTROL PLANE

In this position paper, we focus on the control plane of the P4P framework, as this is an area that IETF effort can clearly improve interoperability. Figure 1 shows the potential entities in the P4P framework: *iTrackers* owned by individual network providers, appTrackers in P2P systems, and P2P clients (or peers for short). Note that there may exist interactions between ISPs and *iTrackers* as well as interactions among iTrackers; however, they are not shown in the figure. Note also that not all entities might interact in a given setting. For example, trackerless systems do not have appTrackers. P4P does not dictate the exact information flow, but rather provides only a common messaging framework to ensure extensibility. A specification of current P4P interfaces in WSDL is shared within P4P Working Group.

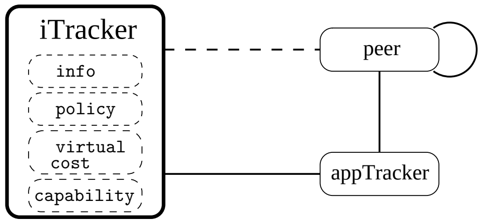

Figure 1: *iTracker* **interfaces and information flow (only** *iTracker* **and P2P are shown).**

3.1 *iTracker* interfaces and functions

The key component of the P4P framework is iTrackers. *iTrackers* provide three interfaces that others can query.

The `info` interface allows others, typically peers inside the provider network, to obtain network topology and status. Specifically, given a query for an IP address inside the network, the interface maps the IP address to a `(ASID, PID, LOC)` tuple, where `ASID` is the ID of the network provider (*e.g.*, its AS number), `PID` is an opaque ID assigned to a group of network nodes, and `LOC` is a virtual or geographical coordinate of the node. Note that the opaque `PID` is used to preserve provider privacy at a coarse grain (*e.g.*, a network provider can assign two `PID`s to nodes at the same point of presence or PoP). Note also that `LOC` can be used to compute network proximity, which can be helpful in choosing peers. When sending an `info` query, a peer may optionally include its swarm ID (*e.g.*, info hash of a torrent). The *iTracker* may keep track of peers participating in a swarm.

The `policy` interface allows others, for example peers or appTrackers, to obtain policies and guidelines of the network. Policies specify how a network provider would like its networks to be utilized at a high level, typically regardless of P2P applications; while guidelines are specific suggestions for P2P to use the network resources. To name a few examples of network policies: (1) traffic ratio balance policy, defining the ratio between inbound and outbound traffic volumes, for interdomain peering links; (2) coarse-grain time-of-day link usage policy, defining the desired usage pattern of specific links (*e.g.*, avoid using links that are congested during peak times); and (3) fine-grain link usage policy. An example of network guidelines is that a network provider computes peering relationships for clusters of peers (*e.g.*, clustered by `PID`). The `policy` interface can also return a set of normalized inter-`PID` costs, which indicate costs incurred to the provider when peers in two PIDs communicate.

The `virtual cost` interface (or `cost` for short) allows others, for example peers or appTrackers, to query virtual network topology and the costs of communications through an ISP's network. In the next section, we will give more detail on this interface.

The `capability` interface allows others, for example peers or content providers (through appTrackers), to request network providers' capabilities. For example, a network provider may provide different classes of services or on-demand servers in its network. Then an appTracker may ask *iTrackers* in popular domains to provide such servers and then use them as peers to accelerate P2P content distribution.

One example of network providers' capabilities is P2P cache servers. Cache discovery is a challenge. The cache providers can inspect the traffic to detect P2P traffic to cache, requiring them to be inline, which is undesirable from the perspective of some ISPs. Alternatively, cache vendors provide proprietary cache discovery protocols, requiring each P2P application to implement and use multiple, proprietary cache discovery protocols, which is inefficient. A design goal of the `capability` interface is that standardized P2P cache discovery is more efficient than multiple proprietary protocols. Specifically, if cache discovery is implemented as a part of the P4P communications (*i.e.*, cache locations are a part of the ISPs' policies and network guidance), then all P2P applications that support P4P will work with all P2P caches that support P4P.

A network provider may choose to implement a subset of the interfaces. The richness of information conveyed is also determined by the network provider. Note that a network provider may also enforce some access control to the interfaces to preserve security and privacy. For example, it may restrict access to only trusted appTrackers.

3.2 Examples

Now we give two examples to illustrate how the *iTracker* interfaces are utilized.

Figure 2 shows an example P2P application with an appTracker using the `policy` and/or the `virtual cost` interfaces to request network policy and/or virtual costs. In the example, a P2P swarm spans two network providers *A* and *B*. Each network provider runs an *iTracker* for its own network. Peer *a* and *b* first register with the appTracker. The appTracker queries *iTracker* A through the interfaces, and then makes peer selection for *a* and *b* considering both application requirements and *iTracker* information. Note that as a variant, assume a trackerless system, then peers will query the interfaces and make local decisions to select its peers. For presentation simplicity, from now on, we focus on tracker-based system.

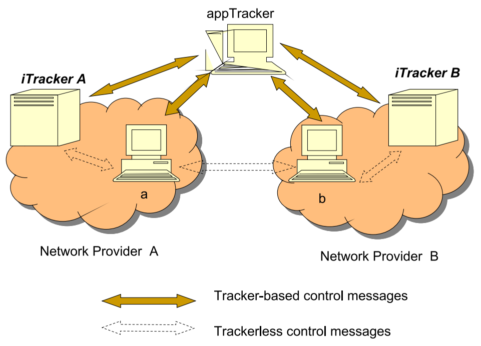

Figure 2: An example of P2P obtaining network policy and cost from portal iTrackers.

Figure 3 shows another example of using P4P. It shows how to request network capabilities through the `capability` interface. Specifically, the appTracker sends a request to *iTracker B* asking the network provider to allocate fixed, high-capacity servers to aid in distributing content. The *iTracker* allocates a server in its network and returns its address to the appTracker. The appTracker will then include the server in returned peer sets for those peers in *B*.

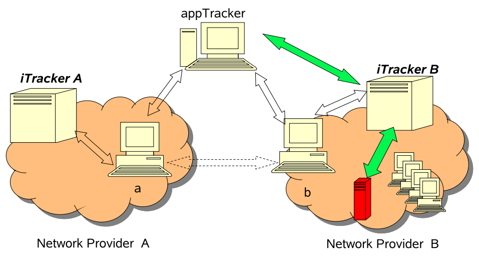

Figure 3: An example of P2P accessing network capability through iTrackers.

4. DECOMPOSITION THROUGH VIRTUAL COST INTERFACE

To support the `virtual cost` interface, the *iTracker* of a network should be connected to the routing system of the network provider. The *iTracker* first constructs an extended network topology $G = (V, E)$ (referred to as virtual topology). This extended topology includes not only the intradomain topology but also one external-domain node for each neighboring domain of the network. Each external-domain node is connected to the internal nodes that correspond to the exit-points by interdomain peering links.

The *iTracker* maps the internal topology G to an external topology $G_k = (V_k, E_k)$ for swarm k. The *iTracker* assigns each node in G_k a unique PID. This mapping may be different for different swarms, but we anticipate that in most cases they will be the same. The objective of the mapping is to preserve ISP privacy, and this can be achieved by scrambling the topology G. A constraint is that the nodes representing the external-domain nodes should be labeled with the autonomous system numbers (ASN).

With virtual topology, a simple peering strategy is local peering. Specifically, when a peer joins a swarm, the appTracker (or the peer in a trackerless system) queries the iTracker. The iTracker can easily locate the joining peer within the ISP's topology using the peer's IP address. It returns the PID of the peer. We refer to peers at PID i as PID-i peers. For simply local peering, when selecting peers of the joining peer, priority should be given to peers with the same PID at the same network. This way, the traffic load across PIDs is minimized. In addition, transport layer connections over low-latency network paths would be more efficient and are therefore desirable from the client's perspective.

A major challenge in designing an interface between *iTrackers* and P2P is that the interface should naturally decompose the overall task into a simple task for the *iTracker* and one for each swarm. By doing so, we are able to achieve an extensible, scalable and efficient design. In such a design, an *iTracker* deals with aggregated P2P traffic, instead of working on the specific detail of a particular P2P application; P2P deals with simple ISP feedback without the need to know the detail of ISP objectives.

A particular novelty of our interface design is that we decouple the algorithms of ISP and P2P using the principle of primal-dual decomposition. This principled interface design leads to a desirable interface. For example, only minor modifications are necessary for computing virtual costs for many different ISP objectives. Please refer to [17] for technical details.

5. EVALUATIONS

We have collaborated with members of P4P Working Group to conduct large-scale Internet experiments. In particular, Verizon provided us with its network topology and Telefonica provided its Peru network topology, and Pando Networks integrated its P2P system with our P4P framework. Pando Networks set up two parallel swarms, each of which had approximately the same number of clients with similar network and geographical distribution. Clients in both swarms shared a 20MB video file. One of the two swarms was P4P-enabled, the other used BitTorrent-like peering.

We observe that the download completion time improves approximately 20% on average. The improvement of download rates for data transfers among Fiber-to-Home (FTTH) clients is 205% on average, and some FTTH clients see as high as 600% improvement.

We also observe that the average number of backbone links in Verizon network used by data delivery dropped from 5.5 to 0.89 on average. In addition, the total external (peering) link load dropped by 70% (outbound) and 53% (inbound), and the total internal backbone link load dropped by 71% (after traffic normalization).

6. CONCLUSIONS

We presented P4P, a simple and flexible framework to enable explicit cooperation between P2P and network providers. It addresses the following issues: (1) explicit integration of network servers or caches to reduce network load (the `capabilities` interface); (2) information from ISPs to applications to signal network bandwidth constraints and policies to optimize P2P network topology (the `virtual cost` interface); (3) enabling applications to signal their bandwidth and priority to networks (the `capabilities` interface and the data plane).

The P4P Working Group, led by Doug Pasko (Verizon; P4PWG co-chair), Laird Popkin (Pando Networks; P4PWG co-chair), and Marty Lafferty (DCIA) made it possible to conduct preliminary field tests on key perspectives of P4P. The results demonstrate that P4P can be a promising approach to improve both application performance and provider efficiency. The working group makes joint efforts towards standardizing the P4P framework and proposing best practices for P2Ps and ISPs. The P4P Working Group consists of 70+ ISPs, P2P companies and researchers working together to address a wide range of business and policy issues. We would encourage the IETF to establish a workgroup to work collaboratively with the P4P Working Group to pursue standards in this area.

7. REFERENCES

[1] Cisco. Network-based application recognition (NBAR). `www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/%122t8/dtnbarad.htm`.

[2] F5 White Paper. Bandwidth management for peer-to-peer applications. `http://www.f5.com/solutions/technology/rateshaping_wp.html`, Jan. 2006.

[3] D. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for internet multihoming. In *Proceedings of ACM SIGCOMM '04*, Portland, OR, August 2004.

[4] S. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer VoIP system. In *Proc of IPTPS*, Santa Barbara, CA, Feb. 2006.

[5] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of SOSP '03*, Bolton Landing, Oct. 2003.

[6] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proceedings of the Internet Measurement Conference*, Berkeley, CA, Oct. 2005.

[7] R. Keralapura, N. Taft, C.-N. Chuah, and G. Iannaccone. Can ISPs take the heat from overlay networks? In *Proc. of HotNets-III*, San Diego, CA, Nov. 2004.

[8] Lightreading.com. P2P plagues service providers.

[9] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proc. of OSDI*, Seattle, WA, 2006.

[10] A. Parker. The true picture of peer-to-peer filesharing. `http://www.cachelogic.com`, July 2004.

[11] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *Proc. of NSDI*, 2007.

[12] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. Selfish routing in Internet-like environments. In *Proc of SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[13] S. Seetharaman and M. Ammar. Characterizing and mitigating inter-domain policy violations in overlay routes. In *Proc of ICNP*, 2006.

[14] G. Shen, Y. Wang, Y. Xiong, B. Y. Zhao, and Z.-L. Zhang. HPTP: Relieving the tension between ISPs and P2P. In *Proc of IPTPS*, Bellevue, WA, Feb. 2007.

[15] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak. Cache replacement policies revisited: The case of p2p traffic. In *Proc of GP2P*, Chicago, IL, Apr. 2004.

[16] H. Xie and Y. R. Yang. A measurement-based study of the skype peer-to-peer VoIP performance. In *Proc of IPTPS*, Bellevue, WA, Feb. 2007.

[17] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provide portal for (P2P) applications. In *Proceedings of ACM SIGCOMM*, Seattle, WA, August 2008.

[18] ZDNet News. ISPs see costs of file sharing rise. `http://news.zdnet.com/2100-9584_22-1009456.html`, May 2003.