P4P: Provider Portal for Applications

Haiyong Xie[†]   Y. Richard Yang[†]
Arvind Krishnamurthy[*]   Yanbin Liu[§]   Avi Silberschatz[†]

[†]Yale University   [*]University of Washington   [§]IBM T.J. Watson

ABSTRACT

As peer-to-peer (P2P) emerges as a major paradigm for scalable network application design, it also exposes significant new challenges in achieving efficient and fair utilization of Internet network resources. Being largely network-oblivious, many P2P applications may lead to inefficient network resource usage and/or low application performance. In this paper, we propose a simple architecture called P4P to allow for more effective cooperative traffic control between applications and network providers. We conducted extensive simulations and real-life experiments on the Internet to demonstrate the feasibility and effectiveness of P4P. Our experiments demonstrated that P4P either improves or maintains the same level of application performance of native P2P applications, while, at the same time, it substantially reduces network provider cost compared with either native or latency-based localized P2P applications.

Categories and Subject Descriptors: C.2.1 [Computer Communication Networks]: Network Architecture and Design – *Network communications*; C.2.3 [Computer Communication Networks]: Network Architecture and Design – *Network Operations*

General Terms: Design, Economics, Management.

Keywords: Network Architecture, Network Application, P2P

1. INTRODUCTION

This paper focuses on the Internet traffic control problem – how network applications (*i.e.*, network resource consumers) efficiently and fairly utilize the network resources owned by network providers. This problem is particularly important as it can have significant impacts on network efficiency, network economics, and application performance.

In the current Internet, traffic control is largely the responsibility of only the network providers (*i.e.*, Internet service providers or ISPs). Applications specify only the destinations of traffic; network providers use optimal traffic engineering to determine efficient routing and satisfy economical objectives such as implementing valley-free routing or multihoming load distribution; network providers can also regulate the transmission rates of applications by controlling network feedback to TCP.

The emerging P2P applications, however, expose significant new challenges to Internet traffic control. Given that a P2P client interested in a piece of data can download it from a number of locations, there is much flexibility in choosing the data sources. This flexibility is one of the key factors contributing to the robustness and scalability of the P2P paradigm. However, this flexibility also

fundamentally changes the network traffic control problem: in the traditional setting, the traffic control problem is typically solved in the context of a given traffic demand pattern; in the new setting, there are multiple ways of satisfying the data demands of an application, each resulting in a different demand pattern and thereby network efficiency. Being largely network-oblivious, many P2P applications may lead to substantial network inefficiency.

First, for intradomain, the network-oblivious peering strategy of many P2P applications may cause traffic to scatter and unnecessarily traverse multiple links inside a provider's network. For example, in our field tests, we observed on the Verizon network that each P2P bit on average traverses 1,000 miles and takes 5.5 metro-hops, while we can reduce the average metro-hops to 0.89 without degrading application performance.

Second, for interdomain, network-oblivious P2P may generate a significant amount of interdomain transit traffic [12] or relay a substantial amount of traffic between the providers of a network [30]. In [12], Karagiannis *et al.* studied the behaviors of BitTorrent on a university network. They found that 50%-90% of existing local pieces in active users are downloaded externally. Even for tier-1 ISPs who do not make payments to network providers, P2P traffic may cause traffic imbalance with its peers, leading to potential violation of peering agreements. Such inefficiency in interdomain traffic may lead to serious disruption to ISP economics.

Third, P2P's dynamic traffic distribution patterns do not necessarily enjoy a synergistic coexistence with network traffic engineering [13, 24] – network providers go to great lengths to estimate traffic patterns and determine routing based on them, but all of this effort could be negated if P2P applications adapt their traffic to changes in the network, thereby resulting in potential oscillations in traffic patterns and sub-optimal routing decisions.

Given the aforementioned issues, both network providers and P2P applications have considered various alternatives. For example, network providers have experimented with multiple traffic control techniques including charging and rate throttling. Unfortunately, none of them appear to be fully satisfactory – without P2P cooperation, the new techniques are ineffective in improving network efficiency and may significantly degrade P2P performance. On the other hand, P2P applications have also unilaterally tried to improve network efficiency by utilizing peering flexibility. For example, several popular P2P applications such as Joost [9] and Kontiki [14] strive to localize application-level peering within the same autonomous system. However, there are fundamental limits on what P2P can achieve alone: to improve network efficiency, P2P applications will have to rely on inferring various types of network information such as topology, congestion status, cost, and policies. Reverse engineering of such information, in particular cost and policy information, is challenging if not impossible.

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

In this paper, we propose a simple architecture called P4P, which stands for provider portal for applications. P4P provides multiple interfaces for networks to communicate with applications regarding: (a) static network policy; (b) p4p distances reflecting network policy and network status, and (c) network capabilities. The interfaces preserve network provider privacy and allow network providers and applications to jointly optimize their respective performance.

At the core of this architecture is the `p4p-distance` interface, through which a network provider can communicate to applications the current "application costs" on its intradomain and interdomain links. We also refer to the interface as the `p-distance` interface for short. The p-distances reflect the network's status and preferences regarding application traffic, and can be used to capture a number of interesting network metrics, such as peak backbone utilization and preferred interdomain links. Applications use these distances to shape their connectivity and choose network-efficient communication patterns if possible.

The `p4p-distance` interface is derived from the principle of optimization decomposition to achieve extensibility, scalability and efficiency. In practice, the interface allows simple and flexible usage. Networks can set the p-distances in a wide variety of ways: from simply ranking preferences, to using OSPF weights, and to computing the shadow prices using primal-dual decomposition. Applications can use the p-distances in a variety of ways as well: from interpreting them as ranks, to using them in sophisticated application-level optimizations. Neither do networks need to know the specifics of applications, nor do applications need to know the specifics of networks or other applications sharing network resources. **Issues**: There are remaining challenges facing P4P. We mention these up front so that researchers can be aware of them.

A major challenge is that both the problem space and the solution space are large and evolving. To demonstrate its flexibility, we have already integrated P4P with multiple representative P2P applications, including BitTorrent (file sharing), Liveswarms (a streaming application), Maze (a Napster-like file distributor), and Pando (a commercial P2P vendor). Many more P2P applications may appear, and thus, we anticipate the specific interfaces presented in this paper to evolve. However, we believe that our proposal of using explicit portals for network providers and the interfaces that they export to be fundamental and valuable. To address the concern of evolving interfaces, we intentionally design the interfaces to be simple and extensible.

Another challenge regarding the P4P architecture is whether network providers and applications have incentives to adopt it. Our extensive evaluations demonstrate the benefits of P4P to both network providers and applications. Recognizing the potential benefits, an industrial working group on P4P has been formed. Its core group consists of ISPs, networking equipment vendors, and P2P software vendors. Although this is certainly encouraging, whether or not this will lead to wide deployment of P4P may take years to know.

2. A MOTIVATING EXAMPLE

We have designed the P4P architecture to be applicable to general settings (not for just P2P) and to support various high-bandwidth applications such as server selection (*e.g.*, mirror site selection), high-bandwidth remote backup, and on-line gaming. In this paper, for concreteness, we focus on P2P.

We use an illustrative example to motivate the need for P4P. Figure 1 shows the Abilene network. The figure also charts the sites of US educational institutions that host PlanetLab installations. These sites typically use Abilene to communicate with each other. Consider a BitTorrent-like P2P application that uses random peering. A client located on the west coast, within the Abilene network, may select many of its peers from the east coast even when there are many Abilene peers available on the west coast. Such long distance peering is not only inefficient to the network but might also
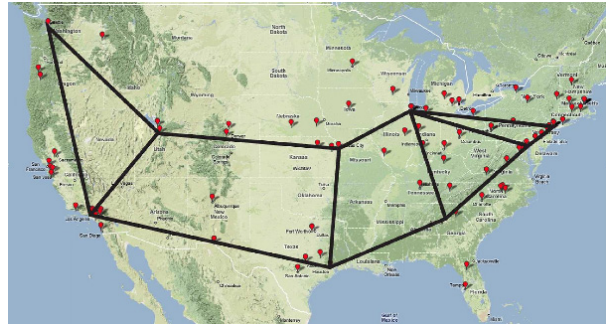

Figure 1: Abilene backbone and PlanetLab sites using Abilene.
lead to lower application performance. Clearly, this inefficiency could be avoided by choosing low-latency or small hop-count peerings, and the resulting locality could reduce the network load as each connection would traverse fewer backbone links.

Pure locality-based peering, however, could cause problems. First, consider the high concentration of clients in certain areas such as the northeastern part of US. Locality-based peering could cause traffic concentrated on a few backbone links. Consider, for instance, the link between New York City and Washington, D.C., a link that is likely to be labeled local measured by both latency and hop count. However, this link is also one of the most congested links on Abilene most of the time. An informed peering strategy, especially one that is aided by network-level information, would distribute peerings, for instance, by having New York clients peer with those in Chicago, and D.C. clients peer with those in Atlanta and Chicago.

Second, consider a heterogeneous deployment where a P2P session consists of clients not only in US educational institutions but also in non-educational networks. Using only latency or hop count, a client might choose to peer with clients that are in the same city or nearby cities, but communicate with them through interdomain links, leading to unnecessary transit costs.

Third, consider many settings where interdomain traffic may be inevitable (*e.g.*, for P2P sessions of smaller size; when taking into consideration content availability; or due to robustness concerns). Being network cost-oblivious, pure locality-based peering may direct clients to choose peers through expensive backup providers, leading to higher transit cost to the hosting network.

The key challenge in solving the aforementioned problems of pure locality-based peering is obtaining network information. The network is at the best position to provide such information. Thus, to explore the tremendous flexibility that P2P has in shaping their traffic to improve network efficiency, it is essential that we introduce cooperation between networks and applications.

3. THE P4P ARCHITECTURE

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

Design Overview
P4P consists of a control plane, a management plane, and a data plane. The data plane is optional and includes functions for differentiating and prioritizing application traffic. The objective of the management plane is to monitor the behavior in the control plane. The focus of this paper is on the control plane.

In the control plane, P4P introduces iTrackers as portals operated by network providers. The introduction of iTrackers allows P4P to divide traffic control responsibilities between applications and network providers, and also makes P4P incrementally deployable and extensible.

Specifically, each network 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
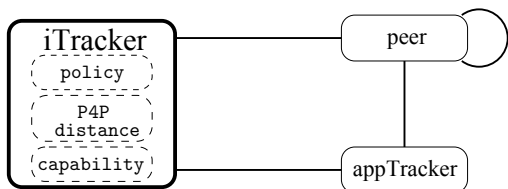
Figure 2: iTracker interfaces and information flow.

iTracker for its network. The iTracker provides a portal for information regarding the network provider. There are various ways to obtain the IP address of the iTracker of a network; one possibility is through DNS query (using DNS SRV with symbolic name `p4p`). Standard techniques can be applied to allow for multiple iTrackers in a given domain, especially for fault tolerance and scalability.

Figure 2 shows the potential entities and information flow in the P4P control plane, in the context of P2P applications: iTrackers representing individual network providers, appTrackers of P2P, and P2P clients (or peers for short). Not all entities might interact in a given setting. For example, in tracker-based P2P, appTrackers interact with iTrackers and distribute the P4P control plane information to peers, while in trackerless P2P that does not have central appTrackers but depends on mechanisms such as DHT, peers obtain the necessary information directly from iTrackers. In both cases, peers can also help the information distribution (*e.g.*, via gossips). An iTracker may be run by a trusted third party instead of by a network provider itself. There also can be an integrator that aggregates the information from multiple iTrackers to interact with applications. P4P does not dictate the exact information flow, but rather provides a common messaging framework.

Example iTracker Interfaces
We give the following example interfaces provided by iTrackers.

The `policy` interface allows applications to obtain the usage policies of a network. To name two examples of network usage policies: (a) coarse-grained 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 (b) near congestion and heavy-usage thresholds, as defined in the Comcast field test [37].

The `p4p-distance` interface allows others to query costs and distance between peers according to networks. In the next section, we will give more detail on this interface.

The `capability` interface allows others, for example peers or content providers, to request network providers' capabilities. For example, a network provider may provide different classes of services, on-demand servers, or caches in its network. An appTracker may query iTrackers in popular domains for on-demand servers or caches that can help accelerate P2P content distribution.

A network provider may choose to implement a subset of the interfaces as determined by the network provider. A network provider may also enforce some access control to the interfaces to preserve security and privacy. For example, a deployment model can be that ISPs restrict access to only trusted appTrackers or information integrators. A provider may also conduct access control for some contents (*e.g.*, for example for the `capability` interface) to avoid being involved in the distribution of certain content.

An Example
Now we give an example to illustrate how the iTracker interfaces are utilized. Figure 3 shows an example P2P application with an appTracker using the `policy` and/or the `p4p-distance` interfaces to request network policy and/or 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 first registers with the appTracker. The appTracker queries iTracker A through the interfaces, and makes peer selection for a considering both application requirements and iTracker information. As a variant, assume a trackerless system. Then peer a will query the interfaces and make local decisions to select its peers. For presentation simplicity, from now on, we focus on tracker-based applications.
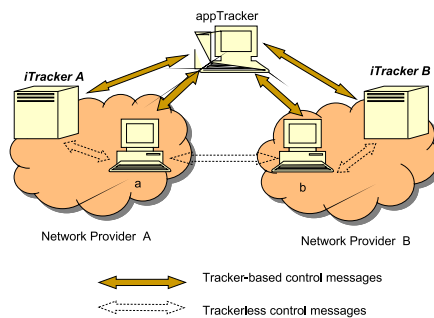

Figure 3: An example of P2P obtaining network policy and p-distances from portal iTrackers.

4. DECOMPOSITION VIA P-DISTANCE

P4P's effectiveness depends on what information is communicated between iTrackers and applications. In this section we present the `p4p-distance` interface and show how it can be used by both ISPs and applications.

Design Requirements
We first state the following design requirements.

- Simplicity and intuitive interpretation. Since network operators and application developers may operate in different settings, the interface has an internal view that is more intuitive to network operators, and an external view for application developers.

- Both network and application control. Although networks are at the best position to provide certain information, applications make end-to-end decisions on how to best utilize all available information, considering both applications and ISPs. Network information should be fine-grained enough to have rich semantics, allowing applications to conduct effective optimizations.

- Extensibility, modularity, and neutrality. Network information should be application-agnostic. An iTracker deals with aggregated traffic, instead of optimizing for a particular application. The information should be in a format that is easy for ISPs to prove, and independent applications to verify, that the ISPs are neutral. Applications deal with simple network feedback, without the need to know network specific details/objectives.

- Scalability. Network information should be aggregated and allow caching to avoid handling per client query to networks. Local/cached information should be useful during both client initialization and application re-optimization.

- Privacy preservation. Network information should be aggregated (*e.g.*, spatial, time, status, and policy) or transformed (*e.g.*, interdomain link cost to congestion level due to virtual capacity). From the application side, query to the network should avoid revealing information about individual clients. This protects the privacy of not only the individual clients but also the client base of an application vendor.

The P4P-Distance Interface
Given the preceding requirements, we now present more details on the interface. The interface has two views: the internal view seen by an iTracker, and the external view seen by applications.

The internal view of the iTracker is a network topology $G = (V, E)$, where V is a set of nodes and E is a set of links. A node in V is referred to as a PID (opaque ID). There are multiple types of PIDs. The first type of PID is an aggregation node, which represents a set of clients. For instance, one aggregation PID may represent a network point of presence (PoP). It is also possible that an aggregation PID may represent the set of clients with a given network status (*e.g.*, the same level of congestion). We call an aggregation PID an externally visible PID. A client queries the network (*e.g.*, iTracker or a provisioning system) to map its IP address to its PID and AS number. This mapping is done when a client first obtains its IP address. If the mapping from IP to PID can be

dynamic, the client may refresh the mapping periodically. For simplicity, we focus on the case that an aggregation PID represents a PoP and is static. To help with configuration and privacy, the internal view may also include PIDs representing core routers and external-domain nodes. These PIDs are not visible by applications and clients. The iTracker connects two PIDs in the internal view if it is meaningful (*e.g.*, each PID represents a PoP and there are physical links connecting the two PoPs). The iTracker assigns a p-distance on each such PID-level link.

The external view of an iTracker to the applications is a full mesh, although an application may be interested only in a subset of the pairs. Given a pair of externally visible PIDs i and j, the iTracker reveals the p-distance p_{ij} from PID-i to PID-j. This p-distance is computed from the network internal distances and routing. An iTracker may perturb the distances to enhance privacy.

ISP Use Cases

The `p4p-distance` interface is a simple and flexible interface. We now present use cases by ISPs and applications.

An ISP can assign p-distances in a wide variety of ways.

- It derives p-distances from OSPF weights and BGP preferences.

- It assigns higher p-distances to links with higher financial costs or approaching congestion (*i.e.*, congestion-level notification).

- It computes p-distances by using the dual (shadow price) variables of optimization decomposition (see Section 5).

There are several dimensions for an ISP to control the distribution of the p-distance information. One dimension is the PID aggregation level. The finest PID level is that each PID is the IP address of an individual client. The advantage of this usage is that it allows fine-grained control. However, it has two problems: (1) it may have scalability issues, as the iTracker now needs to answer per-client queries; and (2) it may have client privacy issues, as the query may reveal too much individual client information.

Another dimension is the granularity and semantics of network information. A "coarsest" level is that given PID-i, the ISP ranks all PIDs, and assigns the most preferred PID p-distance 1, next 2, and so on. This scenario is simple and may have better robustness. The major issue, however, is that ranking is coarse-grained (*e.g.*, the second ranked may be as good as the first one or much worse), and thus does not allow precise traffic control (*e.g.*, 20% to PID-1 and 80% to PID-2). It also has weak semantics. Assume that the ranking is PID-1 over PID-2 over PID-3. Then it is unclear how to compare two sets, when a set consists of clients from multiple PIDs. For example, one set consists of one client from PID-1 and three from PID-3, while the other consists of four clients from PID-2. Such sets arise when applications build structures (*e.g.*, a spanning tree). It is difficult to use ranking in many application optimization formulations.

The third dimension is the recipient of the information. An ISP can reveal more precise information to trusted recipients. This improves ISP security and privacy but may lead to weaker neutrality.

The preceding discussions are examples of tradeoffs that may be made in using the interface. These tradeoffs involve scalability, richness of semantics, and privacy. The interface is simple, flexible, and standardized to allow inter-operation. The usage, which determines complexity, will be chosen by the ISPs individually.

Application Use Cases

Applications can use the interface in a variety of ways. They can combine the p-distance map with performance maps (*e.g.*, delay, bandwidth or loss-rate) to make application decisions. Performance maps can be obtained from ISPs or third parties. Applications may set lower rates or back off before using higher p-distance paths. Below we focus on using the interface for P2P peer selection. We refer to a P2P client at PID-i as a PID-i client or peer.

P-Distance as Ranks
One simple peering strategy is that the probability of a PID-i client selecting a peer at PID-j should be a decreasing function of p_{ij}. In the extreme, the P2P application can consider the p-distances as ranks: the lower the distance, the higher the rank. Specifically, when a client from PID-i joins a P2P session, the appTracker (or the peer in a trackerless system) queries the iTracker to obtain the p-distances to other PIDs from PID-i. If the iTracker assigns p-distance to be low inside PID-i, higher to other PIDs, and even higher to external networks, then using the p-distances as ranks, the application reduces traffic load across PIDs and autonomous systems. In addition, transport layer connections over low-latency network paths would be more efficient and are therefore desirable to clients.

Black-box Peer Selection
Issues of using the p-distances as ranks include load balancing and weak robustness. Also, when selecting peers for a client, many P2P applications build certain structures (*e.g.*, spanning tree or other complex structures) to achieve certain connectivity. If such an application has a random component, then, instead of running the peer selection algorithm only once, the application can run it multiple times. It compares the total p-distances of the multiple runs, and selects the one with the lowest value.

Application with Upload/Download Matching
Some applications may want to maximize the matching of peer download and upload (see, *e.g.*, [23]). Assume P2P session k computes that the PID-i peers have u_i^k and d_i^k total uploading (supply) and downloading (demand) capacity to peers in other PIDs. Let t_{ij}^k be the traffic volume from PID-i to PID-j by P2P session k. Without considering network efficiency, the session may want to optimize:

$$\max \quad \Sigma_i \Sigma_{j \neq i} t_{ij}^k \tag{1}$$

$$s.t. \quad \forall \text{ PID } i, \Sigma_{j \neq i} t_{ij}^k \leq u_i^k, \tag{2}$$

$$\forall \text{ PID } i, \Sigma_{j \neq i} t_{ji}^k \leq d_i^k, \tag{3}$$

$$\forall i \neq j, t_{ij}^k \geq 0. \tag{4}$$

For each PID-i, (2) is aggregated uploading capacity constraint, and (3) is aggregated downloading capacity constraint.

Considering ISP objective, a session k may choose to optimize network efficiency so long it achieves at least β of the preceding optimal solution, *OPT*, where β is an efficiency factor (*e.g.*, 0.8). Thus, the objective of session k is to minimize:

$$\min \quad \Sigma_i \Sigma_{j \neq i} p_{ij} t_{ij}^k \tag{5}$$

under the constraints of (2)-(4) as well as the following:

$$\sum_i \sum_{j \neq i} t_{ij}^k \geq \beta * OPT. \tag{6}$$

Application with Robustness Constraint
To avoid the case that considering ISP objective leads to lower robustness, clients in a given PID may need to connect to a minimum number of peers in other PIDs. One possibility is that the session specifies such preferences by introducing $\underline{\rho}_{ij}^k$ as a lower bound on the percentage of traffic from PID-i clients to PID-j clients among the total traffic from PID-i clients to clients in all other PIDs. Note that $0 \leq \underline{\rho}_{ij}^k \leq 1$, and $\forall i, \Sigma_{j \neq i} \underline{\rho}_{ij}^k < 1$. If PID-$i$ clients prefer downloading from PID-j clients, because, *e.g.*, the latter have more desirable chunks, then $\underline{\rho}_{ij}^k$ should be relatively larger. Then besides (6), the appTracker has an additional robustness constraint:

$$\forall i, j \neq i, t_{ij}^k \geq \underline{\rho}_{ij}^k \sum_{j' \neq i} t_{ij'}^k. \tag{7}$$

5. P4P-DISTANCE AS AN OPTIMIZATION DECOMPOSITION INTERFACE

The preceding section gives use cases of the `p4p-distance` interface. In this section, we present the theoretical foundation behind the interface design.

Foundation

We first introduce some notations. The iTracker collects network status information including (1) b_e, the amount of background traffic on edge e (*i.e.*, traffic not controlled by P4P), (2) c_e, the capacity of edge e, and (3) $I_e(i, j)$, the indicator of edge e being on the route from PID i to j in the topology G. Let T^k be the set of acceptable traffic demand according to the requirements and properties of application session k. Consider one acceptable $t^k \in T^k$. If application session k chooses t^k, it will generate t_{ij}^k amount of peering traffic from PID i to PID j. Correspondingly, let t_e^k be the amount of traffic on link e.

For concreteness, we present the case of traditional ISP traffic engineering objective: to minimize the maximum link utilization (MLU).

$$\min_{\forall k: t^k \in T^k} \max_{e \in E} \quad (b_e + \sum_k \sum_i \sum_{j \neq i} t_{ij}^k I_e(i, j))/c_e$$

Figure 4: ISP using MLU as objective.

A centralized solution to the problem would require the iTracker and each application session to share all information, which is infeasible. To decompose the problem and develop a distributed solution, we consider the following rewriting of the ISP objective:

$$\min_{\alpha, \forall k: t^k \in T^k} \alpha \tag{8}$$

$$s.t. \quad \forall e \in E : b_e + \sum_k t_e^k \leq \alpha c_e. \tag{9}$$

Introducing a dual variable $p_e \geq 0, e \in E$ for each constraint of (9), we define the Lagrange dual function

$$D(\{p_e\}) = \min_{\alpha, \forall k: t^k \in T^k} \alpha + \sum_e p_e(b_e + \sum_k t_e^k - \alpha c_e). \tag{10}$$

To make $D(\{p_e\})$ finite, we need the coefficient of α in $D(\{p_e\})$ to be zero:

$$\sum_e p_e c_e = 1. \tag{11}$$

Then we simplify $D(\{p_e\})$ to:

$$D(\{p_e\}) = \min_{\forall k: t^k \in T^k} \sum_e p_e(b_e + \sum_k t_e^k) \tag{12}$$

$$= \sum_e p_e b_e + \sum_k \min_{t^k \in T^k} \sum_e p_e t_e^k. \tag{13}$$

A particularly pleasant result of the preceding derivation is that the problem now is naturally decomposed into *independent* problems for individual application sessions! The coupling between the iTracker and the applications is also decoupled. In other words, the objective of each application session is to pick t^k among the set T^k of all acceptable traffic pattern, so that $\sum_e p_e t_e^k$ is minimized. After that, the iTracker adjusts $\{p_e\}$. As we will show, this natural decomposition is valid not only for MLU, but also for several other common ISP objectives.

Interactions

With the preceding natural decomposition, the interactions between an iTracker and applications are clean to design. Figure 5 illustrates the interaction structure between an iTracker and applications, in the theoretical framework.

Specifically, application session k obtains $\{p_{ij} | p_{ij} = \sum_e p_e I_e(i, j)\}$ from the iTracker of the ISP, and locally computes $\bar{t}^k$ to optimize
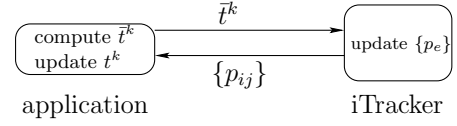

Figure 5: Interactions between iTracker and applications.

$\sum_{ij} p_{ij} t_{ij}^k$. An example of such local optimization is the bandwidth matching example shown in (5). The iTracker gets feedback (*i.e.*, $\bar{t}^k$) from applications, and adjusts $\{p_e\}$ using projected super-gradient method:

$$p_e(\tau + 1) = [p_e(\tau) + \mu(\tau)\xi_e(\tau)]_S^+, \tag{14}$$

where $\xi(\tau) \in \partial D(\{p_e\})$ is a super-gradient, $[x]_S^+$ is projection onto $S = \{p_e | \sum_e c_e p_e = 1, p_e \geq 0\}$, and $\mu(\tau)$ is the step size. Then the iTracker updates $\{p_{ij}\}$ and sends them to any querying application sessions.

To give a concrete example of a super gradient, we have:

PROPOSITION 1. *Let* $S = \{p | \sum_{e \in E} c_e p_e = 1; \forall e \in E, p_e \geq 0\}$ *and* $p \in S$. *Suppose that* $\{\bar{p}\} \in S$ *is given and that* $(\alpha, \{\bar{t}^k\})$ *is an optimal solution of* (12) *at* $\{\bar{p}_e\}$. *Then* $\{\xi | \xi_e = b_e + \sum_k \bar{t}_e^k - \alpha c_e\}$ *is a super-gradient of* $D(\cdot)$ *at* $\{\bar{p}_e\}$.

In practice, application session k may not be able to completely adjust its traffic pattern to $\bar{t}^k(\tau)$, but rather change it to $t^k(\tau + 1) = t^k(\tau) + \theta(\tau)[(\bar{t}^k(\tau) - t^k(\tau)]$, where $\theta(\tau) > 0$ is a step size. Thus, ξ_e can be set to $b_e + \sum_k t_e^k(\tau + 1) - \alpha c_e$, where the sum of the first two terms can be estimated via traffic measurements at edge e. Theoretically, the choices of step sizes ($\mu(\tau)$ and $\theta(\tau)$) are important for the algorithm to converge. In practice, however, since the network and applications continuously evolve, a constant step size may be used.

Extensions to ISP Objective

The preceding derivation is for minimizing MLU. ISPs may have different traffic engineering objectives. We next give three more example ISP objectives. These objectives are proposed by members of the P4P working group. Only minor changes to iTracker are needed to accommodate these objectives. Due to the decoupling, there is no need for application changes.

Bandwidth-Distance Product

An ISP may have a distance metric assigned to each link, denoted by d_e for link e. The end-to-end distance is $d_{ij} = \sum_{e \in path(i,j)} d_e$ from node i to node j. When $d_e = 1$, it de-generates to hop-count distance. The ISP objective could be minimizing the bandwidth-distance product (BDP):

$$\min_{\forall k: t^k \in T^k} \sum_k \sum_i \sum_{j \neq i} d_{ij} t_{ij}^k = \sum_k \sum_e d_e t_e^k$$

$$s.t. \quad \forall e \in E : b_e + \sum_k t_e^k \leq c_e.$$

The dual function is

$$D(\{p_e\}) = \sum_e p_e(b_e - c_e) + \sum_k \min_{t^k \in T^k} \sum_e (p_e + d_e)t_e^k. \tag{15}$$

Then, following the same technique used before, we can derive the rules for iTracker to update and communicate distance $\{p_{ij} + d_{ij}\}$ to applications.

Peak Bandwidth

One ISP requires that they would optimize the MLU (or bandwidth distance product) for the cases when underlying traffic reaches its peak bandwidth usage. By doing so, the ISP could provision more bandwidth and provide better services for background traffic (P2P traffic is deemed as "less-than-best-effort" traffic).

One way of optimizing for peak bandwidth is that the ISP can simply set $\{b_e | b_e = b_e(t_{peak})\}$; that is, we use the background traffic volumes at peak times, in the optimization problems formulated in the preceding subsections. Nothing else needs to be changed.

Another way is that $\{b_e | b_e = \max_t b_e(t)\}$; in other words, the background traffic volume of a link is set to the maximum volume over a certain time period.

Interdomain Multihoming Cost Control

One common request by non-tier-1 ISPs is to control the cost increase due to the increasing P2P traffic.

To precisely state this objective, we use the percentile-based charging model, as this is a typical usage-based charging scheme currently in use by many network providers (*e.g.*, [5, 18, 26]). For example, in the 95th-percentile charging model, an ISP's provider keeps track of traffic volumes generated by the ISP during every 5-minute interval. At the end of a charging period (typically one month), the provider sorts the volumes in ascending order, and charged the ISP based on the traffic volume sent during the 8208-th ($95\% \times 30 \times 24 \times 60/5 = 8208$) sorted interval. This volume is referred to as the charging volume.

We denote by v_e the virtual capacity for P4P-controlled traffic for a interdomain link e. If the amount of P4P-controlled traffic over the interdomain link can be bounded by v_e such that the sum of v_e and background traffic does not exceed the charging volume, then the ISP's financial cost will remain the same.

Now the ISP objective can be reflected by the following constraint on an interdomain link e:

$$\forall \text{ interdomain link } e, \sum_k \sum_i \sum_{j \neq i} t_{ij}^k I_e(i,j) \leq v_e. \tag{16}$$

We add constraint (16) to the problem formulation in Figure 4 to optimize for both intradomain and interdomain objectives.

6. P4P IMPLEMENTATION

We have implemented several iTrackers (for different ISP objectives) and appTrackers (for different P2P applications). We define the P4P interfaces in WSDL, and implement the iTrackers using SOAP toolkits. We focus on the `p4p-distance` interface.

6.1 iTracker

In our implementation, an iTracker can specify either static p-distances or dynamic p-distances that are computed using the projected super-gradient method described in the preceding section. If dynamic, the iTracker collects traffic measurements, and updates its p-distances every T seconds, where T is a parameter.

The update of the intradomain p-distances is relatively straightforward. To update the interdomain multihoming p-distances, the iTracker will need to estimate the virtual capacity v_e available for P4P controlled traffic.

In our implementation, the iTracker first predicts the charging volume of the current charging period in a q-percentile model. Let v be the vector containing all of the 5-minute traffic volumes in a charging period with I intervals, and $\tilde{v}_i$ the predicted charging volume for interval i. Note that a pure sliding window approach (*i.e.*, always use the last I samples for prediction) may not work well. Using the Abilene traffic traces used in [35] and assuming the available virtual capacity in each interval is fully utilized, we found that a pure sliding window approach could result in over-utilization or under-utilization if the charging volume of the preceding charging period was significantly lower or higher than the actual charging volume of the current period.

Instead, we use the following sliding window approach to predict $\tilde{v}_i$ for interval i: we use the last I volume samples for prediction in the first M intervals in a charging period (*i.e.*, when the number of volume samples is not large enough); otherwise, we use all volume samples in the current charging period for prediction:

$$\tilde{v}_i = \begin{cases} \mathtt{qt}(v[i-I, i-1], q) & \text{for} \quad s \leq i \leq s+M, \\ \mathtt{qt}(v[s, i-1], q) & \text{for} \quad s+M < i < s+I, \end{cases}$$

where $s = \lfloor \frac{i}{I} \rfloor * I + 1$ is the first interval in the current charging period, and $\mathtt{qt}(v, q)$ is the q-th percentile value in v.

The iTracker next predicts the traffic volume for the current interval using the moving average of volumes in recent intervals within a sliding window. The size of the sliding window is a parameter of both prediction algorithms. However, it cannot be too large; otherwise, the diurnal traffic patterns may be lost in the prediction.

Finally, the iTracker takes the difference between the predicted charging volume $\tilde{v}_i$ and predicted traffic volume to estimate v_e.

6.2 appTrackers

We have also integrated P4P with the application trackers (appTrackers) of multiple tracker-based P2P applications: BitTorrent (as a representative file sharing application), Liveswarms [22] (as a representative swarm-based real-time streaming application), and Pando [21] (as a representative BitTorrent-like commercial file sharing application). The only change to client software is to collect experimental statistics. We leave the implementation for trackerless applications as future work.

General Issues

First, clients join and leave an existing P2P session dynamically. A new client will contact the appTracker to get its peering neighbors, and an existing client may do so as well due to the departure of its neighbors. In our implementation, the appTracker collects information for these requesting clients and makes decisions for them. On the iTracker's side, the traffic of existing connections in the P2P session is automatically reflected as a part of the background traffic.

Second, consider how to select peers for a client located at PID-i of autonomous system (AS) n. Assume that a client needs m peers. In our implementation, the appTracker selects these m peers for this client in three stages. The objective of the staged peer selection is to provide sufficient connectivity and robustness.

- First, it selects among those that are also located at PID-i. This is called *intra-PID peer selection*. The appTracker selects up to Upper-Bound-IntraPID fraction of the m peers during intra-PID peer selection. The default value of Upper-Bound-IntraPID is 70%. Note that many PIDs may not have a large number of clients. Thus, Upper-Bound-IntraPID mainly serves as an upper bound. The bound will be set to a lower value if the network p-distance within PID-i is relatively higher than outside the PID.

- Second, the appTracker expands the selection to other PIDs but at the same AS-n. This is called *inter-PID peer selection*. The appTracker selects up to Upper-Bound-InterPID fraction of the m peers from AS-n, including those selected during the intra-PID stage. To be valid, Upper-Bound-InterPID should be higher than Upper-Bound-IntraPID. The default value of Upper-Bound-Inter-PID is 80%. It will be set to a lower value if the interdomain p-distances from AS-n are relative lower than those inside AS-n. Below, we give concrete examples on how to use p-distances to guide inter-PID peer selection.

- In the last stage, the appTracker selects from outside the AS to reach a total of m peers. This is called *inter-AS peer selection*. A particular challenge in inter-AS peer selection is that two ASes may have conflicting p-distances on traffic between them (*e.g.*, a provider may prefer connecting to a customer, who may prefer to connect to its customers). One possibility to solve this problem is to use Nash Bargaining Solution. In our implementation, when a client from AS-n joins, the appTracker uses the p-distances from AS-n's view. Thus, the more clients an AS has, the more opportunity the AS has in guiding application traffic according to its p-distances. The fraction of peers selected from AS-n' during inter-AS peer selection is inverse proportional to the p-distance from PID-i to AS-n'.

BitTorrent: Our implementations for BitTorrent-based appTrackers are based on BNBT EasyTracker, a popular tracker for BitTorrent. The native BitTorrent appTracker chooses peers randomly. We implement two additional types of BitTorrent: delay-localized BitTorrent, in which a client chooses peers with lower latency; and

P4P BitTorrent, in which the appTracker periodically obtains p-distances from iTrackers. It uses the following algorithm for inter-PID peer selection. For each $j \neq i$, it computes weight $w_{ij} = \frac{1}{p_{ij}}$, for $p_{ij} \neq 0$; if $p_{ij} = 0$, it sets w_{ij} to be a large value. It normalizes $w_{ij} = \frac{w_{ij}}{\sum_{i \neq j} w_{ij}}$. To increase robustness, it applies a concave transformation on w_{ij} to increase the relative weights of small w_{ij}. This transformation can be considered as a simple implementation of the robustness constraint in (7).

Liveswarms: Liveswarms is a variant of BitTorrent for streaming. Its clients are very similar to BitTorrent clients, but with admission control and resource monitoring to accommodate real-time streaming requirements. P4P Liveswarms inter-PID peer selection is similar to that of P4P BitTorrent.

Pando: Pando is a large BitTorrent-like P2P system with more than 10 million users. Our P4P Pando integration conducts download/upload bandwidth matching optimization shown in (5), for clients inside a given AS. Let t_{ij} be the ideal inter-PID traffic demand distributions computed by solving (5).

One key issue in P4P Pando integration is how to implement t_{ij}. One way to implement the desired traffic distribution t_{ij} is to derive per-peer bandwidth allocations, by taking into account the computed t_{ij} values and the number of peers at each PID. These per-peer bandwidth allocations are communicated to each peer, and each peer has a local connection manager to monitor and enforce these limits. Although some popular P2P client implementations (*e.g.*, Azureus) do have software components for enforcing such limits, this approach requires tighter integration and more pervasive changes to client software; more importantly, such integration requires that Pando re-architecture its production system.

Instead, our implementation takes an approach that trades off strict adherence to the ideal traffic distribution for simplicity, by mapping the desired P2P demand to how clients should establish peering relationships with each other. Specifically, once the t_{ij} values have been computed, we derive the following weights $w_{ij} = t_{ij} / \sum_j t_{ij}$, where $i \neq j$. Similar to P4P BitTorrent, P4P Pando increases the relative weights of small w_{ij} to increase robustness. Then, when a client at PID i is faced with the decision of choosing peers, it should pick peers at PID-j with probability w_{ij}. Note that this scheme operates at a coarser grain than the alternative of enforcing per-connection limits. Also note that our implementation does not achieve strict bounds on traffic between underlying nodes; instead, it controls only the number of connections in a probabilistic manner. As mentioned earlier, we favor this approach given its simplicity and ease of integration to client software.

The remaining issue is who should conduct the bandwidth matching optimization. The input to this optimization is estimates of client up/download bandwidth and p-distances from the iTracker. To minimize the complexity and risks of integration of the Pando production system, we provide a middleware service called appTracker Optimization Service. This service runs between the Pando appTracker and iTrackers. In particular, the Pando appTracker periodically queries the service. The query includes Pando's estimates of client up/download bandwidth. The appTracker Optimization Service takes these estimates, queries the iTrackers, conducts optimization, and returns the weights to Pando appTracker. The Pando appTracker uses these weights to make peering decisions for clients. We believe that this middleware service deployment model has many merits and may have value in many settings.

7. EVALUATIONS

We have evaluated the effectiveness of P4P using both simulations and real Internet experiments on PlanetLab. In addition, we have conducted large-scale test deployment with a major commercial P2P vendor, Pando Networks, Inc. on networks of major ISPs. Our results show that P4P can improve not only network efficiency

but also P2P application performance. Below we focus our evaluations on how a network provider and peers can effectively utilize the `p4p-distance` interface.

7.1 Evaluation Methodology

Network Topologies: We evaluate P4P using real network topologies. Table 1 summarizes a subset of the topologies. We conduct real Internet experiments on Abilene and ISP-B. We conduct simulations on PoP-level topologies of Abilene and major tier-1 ISPs. The PoP-level topologies differ from the real router-level topology, but still illustrate the scope and power of the methods proposed here.

Applications: We report results for BitTorrent, Liveswarms and Pando. In the experiments for each application, we run the corresponding appTracker described in Section 6.2. In all experiments, only the appTrackers are modified to integrate with P4P.

Performance Metrics: We consider the following performance metrics:

- *Completion time:* It measures the performance of BitTorrent and Pando. It is defined as the total time for a swarm of peers to finish downloading a file.

- *P2P bandwidth-distance product:* We refer to the average number of backbone links that a unit of P2P traffic traverses in an ISP's network as unit bandwidth-distance product (or unit BDP for short). We do not run experiments that optimize this objective, but report results using this metric for some experiments to give better intuition. Since the topologies used in the evaluations do not include access link information, we count only backbone links.

- *P2P traffic on top of the most utilized link:* This is the total P2P traffic on the most utilized link in a network. Since in our real experiments, the amount of traffic that we can control is small compared with the overall traffic volume, we use this metric. We also refer to this metric as *P2P bottleneck traffic*.

- *Charging volume:* This metric is only used in interdomain settings. We compute it using the 95-percentile charging model.

Simulations: We build a discrete-event simulation package to simulate BitTorrent. We follow the simulation methodology used in [3] and implement the native BitTorrent protocol. We also model the performance of TCP connections at a session level, as opposed to a packet level, since fine-grained packet level simulations would be infeasible for hundreds of peers sharing a large file. We follow the approach taken in [4] and compute TCP throughput assuming that TCP capacity sharing achieves maxmin fairness in steady state. When there are arrivals or removals of TCP sessions, we recalculate the throughput of affected TCP sessions. We also compute the amount of traffic on each link accordingly to keep statistics of link utilization in the network.

In our simulations, we construct a network to include a given number of peers as follows. We randomly assign each of them to a PoP node in the underlying network. Each peer connects to its assigned PoP through an access link, which has 100 Mbps capacity in both directions. We simulate two equal-sized swarms each sharing a 256 MB file, with block size being 256 KB. Initially each swarm has only one seed with 1 Gbps upstream access link capacity.

Internet Experiments: Besides simulations, we also conduct three sets of real Internet experiments to evaluate P4P. The first set is BitTorrent experiments, where we set up three parallel swarms using PlanetLab nodes to evaluate the performance of P4P against that of native BitTorrent and BitTorrent enhanced with localized peering (with clients connecting to peers based on round-trip delay information). Specifically, we run the iTracker and three separate appTrackers on local hosts, one for each swarm. We also run an initial

Network	Region	Aggregation level	#Nodes	#Links	Usage
Abilene	US	router-level	11	28	Internet experiments, simulation
ISP-A	US	PoP-level	20	-	simulation
ISP-B	US	PoP-level	52	-	Internet experiments
ISP-C	International	PoP-level	37	-	Internet experiments

Table 1: Summary of networks evaluated.

seed server hosting a 12 MB file for each swarm. All three seed servers are configured to have 100 KBps upload bandwidth. They are always co-located in the same PoP, but on different nodes.

We modified the client software to log the activities of block uploading and downloading, and collected delays to all known peers of a swarm once every minute. In all experiments, we randomly choose 160 nodes among a pre-selected set of 187 university nodes. We run a client process for each of the three types of BitTorrent on each of the chosen nodes. Thus each swarm always consists of the same set of clients. All clients join the swarm randomly within a 5-minute period. Note that for P4P, the p-distances before the arrivals reflect pre-arrival network MLU. Such a batch arrival is more challenging as gradual arrivals will give the network more time to adjust.

Each parallel experiment ends when all clients in the swarm complete the download. We run the experiments during late nights when the traffic load on Abilene is relatively light. We also run the experiments multiple times and compute their average.

In P4P BitTorrent experiments, we configure the iTracker to protect a high-utilization link from Washington DC to New York City. The iTracker initially assigns 0 to p-distances, and increases the p-distance of the protected link if clients use this link. The appTracker periodically queries the iTracker to obtain the network p-distances.

The second set is Liveswarms experiments. The setups are similar to BitTorrent experiments, except that we run clients on 53 randomly chosen PlanetLab nodes. The clients form a swarm to stream a 90-minute MPEG video, and each of the experiments lasts 20 minutes.

The third set is field tests using Pando clients, where we set up two parallel swarms to share a popular 20 MB video clip. One swarm uses the native Pando system, and the other uses P4P-integrated system. We set up two iTrackers for ISP-B and ISP-C networks, respectively. These iTrackers compute peering weights as described in the preceding section for appTrackers to select peers in each network. In the experiments, when a Pando client is trying to download the clip, it is randomly assigned to one of the two swarms. The clients report the amount of traffic downloaded from each other peer. We analyze and present the data logged from Feb. 21 to Mar. 2, 2008.

In all experiments, we compute P2P traffic demands for the whole network based on the logs collected by the clients. We then derive bandwidth usage on each link from these demands.

7.2   P4P Intradomain

We evaluate the effectiveness of P4P for intradomain cost control through real Internet experiments on PlanetLab. Since the swarm size is limited by the number of available PlanetLab nodes on Abilene, we also use simulations to study the P4P benefits with a varying number of BitTorrent clients. In these experiments, the ISP objective is to minimize MLU.

Base case: BitTorrent Internet
We start with Internet experiments on PlanetLab using three types of BitTorrent. Figure 6 shows the results. We make the following observations. First, for completion time, native BitTorrent performs poorly in comparison to delay-localized and P4P BitTorrent. Specifically, when compared against Native, P4P results in an improvement ranging from 10% – 20%; delay-localized BitTorrent results in a completion time that is slightly better than P4P.
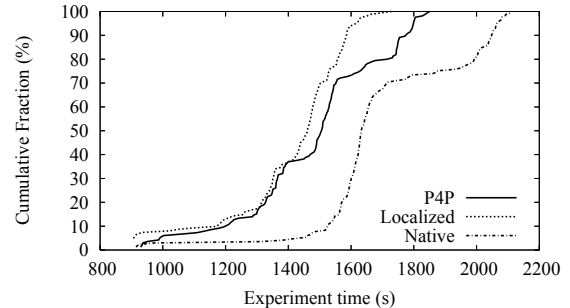
(a) CDFs of completion time.
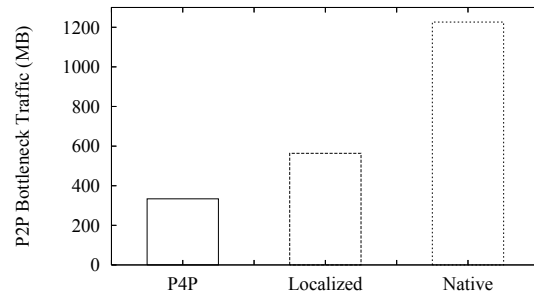
(b) P2P bottleneck traffic.

Figure 6: BitTorrent Internet experiments.

Second, compared against delay-localized and native BitTorrent, P4P significantly reduces P2P bottleneck traffic, achieving ISP objective. Specifically, native BitTorrent results in more than 200% higher traffic volume on the bottleneck link; delay-localized BitTorrent is not aware of the ISP objective, and thus places at least 69% more traffic than P4P on the bottleneck link.

Variation of Swarm Size
Next we use simulation to study the P4P benefits when the swarm size varies. Figures 7 plots the completion time and bottleneck link utilization for Abilene as we vary the number of peers in a swarm sharing a 12 MB file. The peers are randomly placed in the network.

We make the following observations. First, compared with native P2P, P4P improves P2P completion time by approximately 20% on average, cuts down the link utilization by approximately 4 times, and reduces the duration of high traffic load by approximately a half, as peers finish their downloads faster. Thus P4P can reduce P2P traffic intensity on the underlying network dramatically. Second, compared with P4P, delay-localized BitTorrent can result in significantly higher bottleneck link utilization, although it has comparable completion time.

P4P Benefits Consistent across Topologies
Next we show the simulation results to study the P4P benefits when the topology varies. Figure 8 plots the results for ISP-A. Note that the values in Figure 8 are normalized by the maximum value of native BitTorrent. In this experiment, compared with native P2P, P4P reduces completion time by approximately 20%, and reduces the bottleneck link utilization by 2.5 times. Delay-localized BitTorrent improves the completion time by a slightly higher percentage, but its bottleneck link utilization can be higher than 2 times that of P4P. These results suggest that P4P benefits are consistent across network topologies.
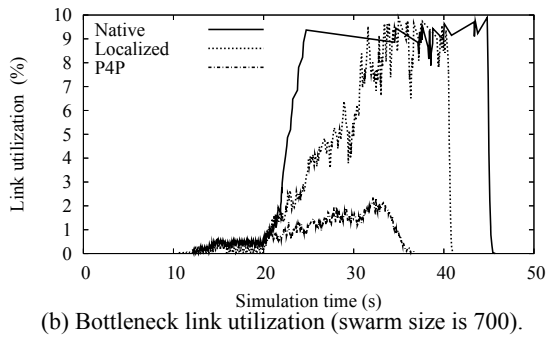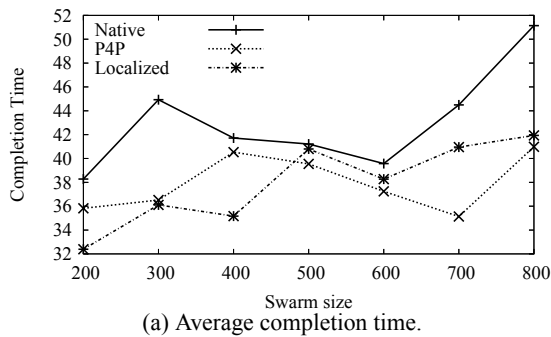
(a) Average completion time.

(b) Bottleneck link utilization (swarm size is 700).

Figure 7: Results on integrating P4P with BitTorrent on Abilene.

(a) Average completion time.

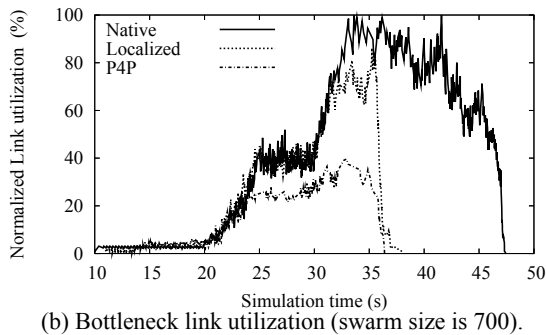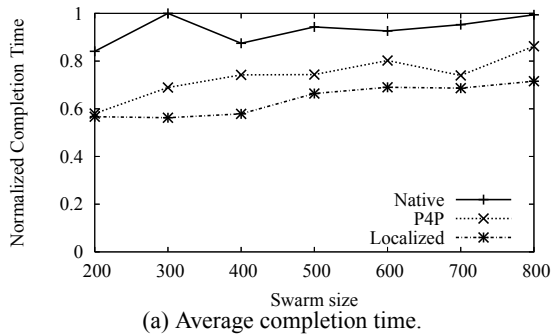(b) Bottleneck link utilization (swarm size is 700).

Figure 8: Results on integrating P4P with BitTorrent on ISP-A.

P4P Robust across P2P Applications

Next we report the evaluation results on integrating P4P with Liveswarms. Our results show that when integrated with P4P, Liveswarms achieves approximately the same level of throughput as without P4P. However, as Figure 9 shows, native Liveswarms results in approximately 50 MB traffic volume on the backbone links on average, while upon integration with P4P reduces the backbone traffic volume to approximately 20 MB. Thus P4P results in approximately 60% reduction on average link traffic volume.
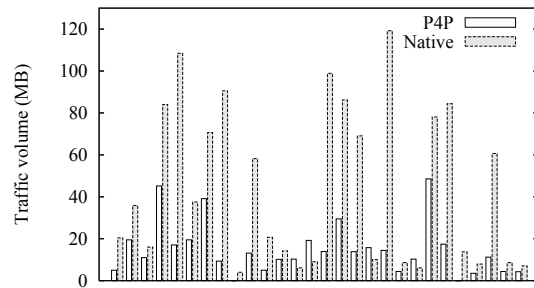

Figure 9: Traffic volumes when integrating P4P with a Liveswarms, a P2P video streaming application.
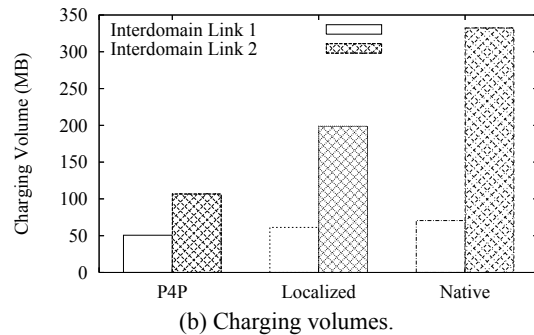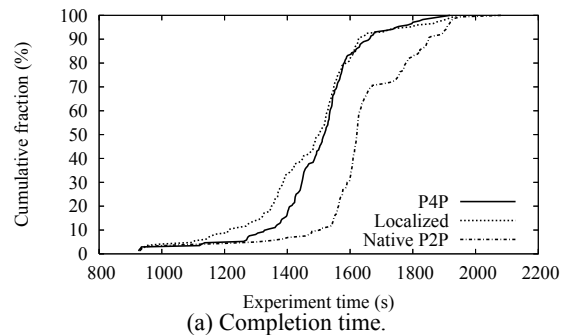

(a) Completion time.

(b) Charging volumes.

Figure 10: BitTorrent Internet interdomain experiments.

7.3 P4P Interdomain Multihoming Cost

Next we study P4P benefits in interdomain multihoming settings. We use PlanetLab nodes to conduct the Internet experiments on Abilene. In the experiments, we take two links in Abilene (the first link between Chicago and Kansas City, and the second between Atlanta and Houston) as two interdomain links, as they partition Abilene into two connected components: one with 4 nodes on the east coast, and the other with 5 nodes on the west coast and midwest. These two components are taken as two "virtual" ISPs. Based on historical traffic volume data in December, 2007 made publicly available by Abilene NOC, we compute virtual P2P capacities for these two links.

Figure 10 shows the results. We make the following observations. First, native BitTorrent leads to a charging volume on the second interdomain link to be as high as 3 times that of P4P due to its peer set selection being oblivious to interdomain policies. Second, delay-localized BitTorrent has slightly better performance on completion time; however, its completion time distribution has a longer tail. Also, its charging volume on the second interdomain link is twice that of P4P; thus it incurs significantly higher bandwidth cost than P4P.

7.4 P4P Field Tests

Next we present the results of P4P field tests. Note that we report ISP-B results only.

Swarm Statistics

Figure 11 plots the dynamics of the numbers of clients in both of the two parallel swarms. We observe that the swarms reached their largest size in the first 3 days, then decreased to and remained at a lower level afterwards. We also observe that the numbers of clients in the two swarms were almost the same, as a result of randomly assigning clients to both swarms. Thus it provides a basis for comparing P4P Pando fairly against native Pando.
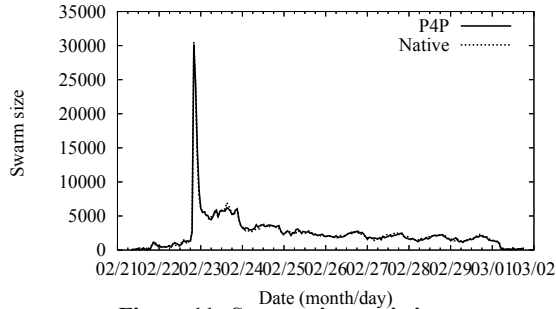

Figure 11: Swarm size statistics.

Total Traffic Statistics

Table 2 summarizes the traffic distribution between ISP-B and the remainder of the Internet. We observe that the download traffic between external clients (outside ISP-B) is approximately the same for Native and P4P, as P4P optimizes for ISP-B clients only. Note that the total amounts of traffic are approximately the same for Native and P4P. However, P4P reduces bandwidth usage on interdomain peering links. In particular, the interdomain traffic volumes flowing into and out of ISP-B generated by Native are 53% and 70% higher than those of P4P, respectively. In addition, P4P results in approximately more than 5 times higher traffic concentration (1 to 0.15) inside ISP-B, as ISP-B clients download more from their intradomain peers.

ISP-B Internal Traffic Statistics

Table 3 summarizes the statistics. We observe that P4P effectively reduces the amount of cross-metro-area traffic within ISP-B network. Specifically, in Native, the amount of intra-metro-area traffic is only 6.27% (of all traffic inside ISP-B); while in P4P, this percentage increases to 57.98%.

Unit BDP

Consider the data transfers among ISP-B clients only. Unit BDP is a good indicator of how far clients download blocks from its neighbors and how much cost the clients incur to the network to some extent. We observe that P4P results in approximately 5 times reduction (from 5.5 to 0.89) in unit BDP, as shown in Figure 12(a). Note that the average number of backbone links between two PIDs in ISP-B is 6.2.

Completion Time

Consider all ISP-B clients. Figure 12(b) shows the completion time distribution. Note that the axis of Figure 12(b) is log scale. Using the distribution, we can calculate that P4P results in an *average* completion time of 7312 seconds, while Native results in 9460 seconds. Thus P4P improves average completion time by 23%.

Note that the ISP-B network topology that the iTracker has does not include the access network information (in particular, last-mile links). Thus, the iTracker does not try to optimize for the access network in the field tests. However, the access network may have significant impact on the completion time and download rates. Therefore, significant improvement in BDP does not always result in as much improvement in the completion time and download rate.

Consider all Fiber-To-The-Premises (FTTP) clients in ISP-B. Figure 12(c) shows the cumulative distribution of FTTP clients' completion time. Using the distribution, we can calculate that P4P results in an average completion time of 2481 seconds, while Native

results in 4164. In other words, the average completion time of Native is 68% higher than P4P for FTTP clients. In addition, a small percentage of FTTP clients can see as much as 300% – 600% improvement on completion time by using P4P.

We note that in the Native experiments, none of the FTTP clients obtain all of the data solely from other FTTP clients in ISP-B. In contrast, in the P4P experiments, most of the FTTP clients obtain all of the data only from other FTTP clients. This is largely resulted by appTracker selecting peers to optimize upload/download bandwidth matching.

8. DISCUSSIONS

Q: [P2P Incentives] It is clear that network providers can benefit from P4P. What are the incentives for P2P to participate in P4P?
A: We give four reasons. First, there are potentials for both network providers and P2P to benefit from adopting P4P. Better network efficiency not only benefits the network providers but also can lead to better application performance, as we clearly demonstrated in our evaluations. Second, as P2P consumes a significant portion of network resources without generating much revenue for providers, there is a real possibility that network providers may limit P2P usage to reduce cost. Thus, in the long run, P2P by playing nice could avoid being blocked or unilaterally rate limited by ISPs. For example, several popular P2P applications such as Joost [9] and Kontiki [14] already consider network efficiency (*e.g.*, localizing peering within ASes) when selecting peers. Third, P4P leaves much flexibility for P2P (*e.g.*, P2P can integrate provider suggestions with its local application-specific requirements). Last, if we consider the Internet as a societal infrastructure, then it benefits the overall society if applications are more network efficient.

Q: [P2P Self Adaptation] Why cannot P2P achieve the benefits of P4P by itself?
A: In the current Internet, for P2P to explore peering flexibility to improve network and application efficiency, it will have to probe the network to reverse engineer information such as topology, and network status. This is however rather challenging in spite of significant progress in network measurement techniques. In addition to the redundant and wasteful network probing that the measurements might cause, a fundamental hurdle is the ability to perform the necessary inferences 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 interdomain policies, traffic balance ratio between peering providers, and 95-percentile based billing).

Without explicit network information, what P2P adapts to by itself, with the objective of helping networks, may turn out to hurt networks. For example, locality of P2P connection is a common self-adaptation objective. However, there are many scenarios where selecting local peers can be less desirable to an ISP. For example, in a wireless network where P2P nodes communicate through a base station, peering with local peers sharing the same base station may require more wireless bandwidth than through the base station to other non-local peers. As another example, a common issue exists in UK is that network providers buy their DSL "last mile" connectivity via a BT central pipe, which provides connection from a DSL customer to its network provider. This connection can be many orders of magnitude more expensive than IP transit. Thus, it can be much more expensive for a network provider to have a customer

	Native	P4P	Ratio (Native : P4P)
External ⟷ External	1,631,202,356,111	1,649,733,748,383	0.99
External ⟶ ISP-B	80,122,658,973	52,324,572,291	1.53
ISP-B ⟶ External	24,585,624,462	14,438,221,829	1.70
ISP-B ⟷ ISP-B	1,747,896,203	11,816,340,071	0.15
Total	1,737,658,535,749	1,728,312,882,574	1.01

Table 2: Overall traffic statistics of field tests: ISP-B.

	Total Traffic	Cross-metro Traffic	Same-metro Traffic	% of Localization
Native	1,747,896,203	1,638,304,110	109,592,093	6.27%
P4P	11,816,340,071	4,965,808,460	6,850,531,611	57.98%
Ratio (Native:P4P)	0.15	0.32	0.02	0.11

Table 3: Internal traffic statistics of field tests: ISP-B.

(a) Average unit BDP.   (b) Completion time.   (c) FTTP completion time.
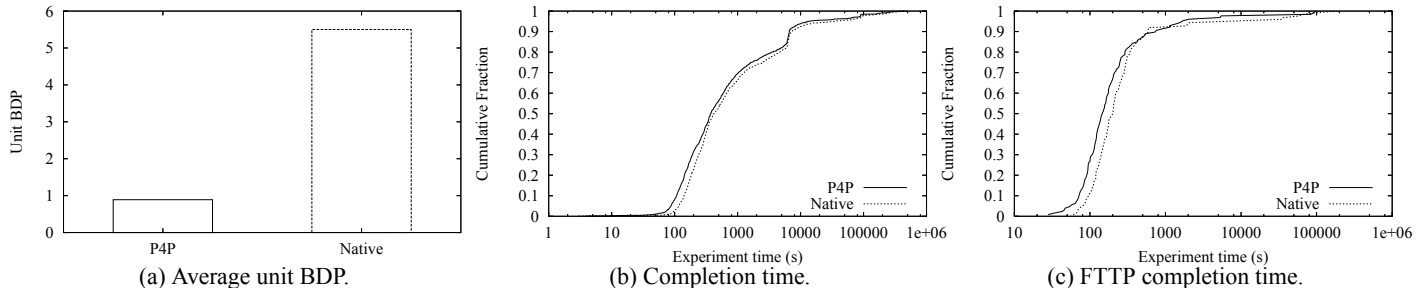
Figure 12: Unit BDP and completion time results of field tests.

retrieve a file from another customer on its network, than it is to go off the network for the file.

Q: [Network Neutrality] Does P4P violate network neutrality?
A: The objective of P4P is to provide a portal for applications and network providers to communicate. The `p4p-distance` interface helps ISPs to demonstrate their neutrality. In our current deployment model, ISPs and P2P applications mutually agree to participate in P4P.

Q: [Scalable Implementation] There can be a large number of P2P networks in a provider's network, and a P2P session may involve a large number of networks. How can it be feasible for P4P to orchestrate all these networks?
A: The iTracker interfaces are light-weight and do not handle per-client application joining request.

In our implementation, an appTracker considers optimizing only the heavy-hitters, namely those networks that comprise of a large number of clients and generate a substantial amount of traffic. If a small number of networks account for a large fraction of traffic, then the appTracker can focus its attention on those networks. In order to quantify to what extent this phenomenon appears in practice, we analyzed the instantaneous swarm behavior of every movie torrent published by `thepiratebay.org`, a popular portal for BitTorrent content. In total, we analyzed 34,721 swarms to determine the number of leechers. We find that only 0.72% of swarms had an excess of hundred leechers. Thus, for most appTrackers, they do not need to maintain states for a large number of ASes. For large swarms spanning many ASes, we could replicate the appTracker and further organize the appTrackers into a two-level hierarchy. The top-level server directs clients to the second-level appTrackers according to the network of the querying client. Another possibility to improve scalability for the `p4p-distance` interface is to use virtual coordinate embedding. This is future work.

Q: [Robustness] A major issue in P2P is to provide robustness. For instance, a BitTorrent client maintains a pool of randomly selected neighbors with which it is just exchanging meta-data information. Do the locality-aware P4P techniques reduce robustness?
A: P4P does not limit the mechanisms for improving robustness. The iTrackers provide only network information, and an appTracker can always select a certain number of random connections to ensure diversity for robustness, as we included in our P2P optimization formulation in Section 4. This diversity typically will not substantially increase the provider cost.

A related robustness feature is that iTrackers are not on the critical path. Thus, if iTrackers are down, P2P applications can still make default application decisions.

Q: [Comparison with Pricing or Priority Schemes] Will multiple levels of traffic priority or non-flat-rate pricing eliminate the need for P4P?
A: P4P will be complementary with such schemes, as these schemes will only impose constraints on the access bandwidth from the edge. They will not resolve the global efficiency problem.

9. RELATED WORK

ISP Approaches: While ISPs have clear incentives to improve network efficiency, existing approaches have limitations. The traditional approach to improving network efficiency is traffic engineering (*e.g.*, [2, 10]). However, an implicit assumption of traditional traffic engineering is that the end-to-end traffic pattern is not fungible, in the sense that one cannot change the source or the destination of a traffic flow. In our project, we consider the fact that the same data may be available from multiple sources and that emerging applications such as P2P may have tremendous flexibility in rewiring their traffic patterns to improve network efficiency.

There are several possibilities on how to control the traffic patterns of P2P by ISPs. One proposal is to deploy P2P caching devices to cut down bandwidth consumed by P2P applications (*e.g.*, [8, 12, 16, 27, 29, 32, 36]). However, P2P caches need to be designed for specific applications and speak the appropriate protocols, which limit their generality and applicability to proprietary protocols. In addition, ISPs may not want to bear the costs of caches if they consider that P2P is a mechanism for content providers to shift distribution costs to ISPs. Furthermore, caching contents may lead to legal liability.

A widely used ISP approach is to use traffic shaping devices to rate limit P2P (*e.g.*, [6, 7, 19, 20, 28, 33]). These devices rely on deep packet inspection or other P2P traffic identification schemes (*e.g.*, [11, 31]). Unilateral rate limiting by ISPs can be considered strong handed and may lead to P2P reactions such as encryption and dynamic ports to avoid being identified. Furthermore, techniques such as rate limiting, end-point usage-based charging, or priority are mainly for controlling edge traffic demands, not for improving network efficiency.

P2P Approaches: A few P2P applications have developed techniques to localize traffic, and there are many techniques for identifying locality (*e.g.*, same AS, low latency, closeness to some landmarks). Several commercial P2P applications also claim using locality in peer selection. Karagiannis *et al.* [12] and Madhyastha *et al.* [17] have observed that locality of P2P connections indeed reduces the download time of users. However, as we discussed, there are many limitations on P2P determining locality. In our project, we target a fundamental solution and leverage the fact that the ISPs are best-positioned to determine locality and to direct applications to not only nearby peers but also to peers that are accessible over well-provisioned links.

There is also previous work on how to design P2P applications that limit their own transfer rates to share network resources with other traffic (*e.g.*, [15, 34]). However, these approaches, similar to the ISP rate limiting approaches, are mainly for controlling traffic demands, instead of for improving network efficiency.

Network Architecture: There are previous proposals on introducing network intermediates and middlewares into the Internet architecture (*e.g.*, [25]). However, we are not aware of any previous proposals on ISP interfaces to achieve higher network efficiency except [1], in which Aggarwal *et al.* independently proposed a scheme in which each ISP provides an oracle. When a new node/peer joins a P2P application, the oracle is presented with a list of IP addresses to be sorted. As we discussed in ISP use cases in Section 4, the design of [1] can be considered as a special case of the `p4p-distance` interface when a PID represents a single IP address and distances are only ranks. Some advantages that the `p4p-distance` style interface has over the IP sorting style include (1) better scalability through aggregation using PIDs; (2) better P2P client and P2P vendor privacy as no individual client information is sent to ISPs; and (3) flexibility, richer semantics, and precision of p-distances over the limited scope of only ranking preferences. We believe that scalability and flexibility are important for Internet architecture design.

10. CONCLUSIONS AND FUTURE WORK

We presented P4P, a simple and flexible framework to enable explicit cooperation between P2P and network providers. Our evaluations demonstrate that it can be a promising approach to improve both application performance and provider efficiency. There are many avenues for further study. In particular, we are conducting more experiments on interdomain traffic control. Improving scalability using virtual coordinate embedding and evaluating the effects of caching are topics under study. Our overall objective is to integrate fairness, congestion management, and efficiency control.

11. ACKNOWLEDGMENTS

12. REFERENCES

[1] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P systems cooperate for improved performance? *ACM CCR*, July 2007.

[2] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc of SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[3] A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and improving a BitTorrent network's performance mechanisms. In *Proceedings of IEEE INFOCOM '06*, Barcelona, Spain, Apr. 2006.

[4] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in Bittorrent via biased neighbor selection. In *Proceedings of IEEE ICDCS '06*, Lisboa, Portugal, July 2006.

[5] S. Cerruti and C. Wright. ISP bandwidth billing. http://www.servicelevel.net/rating_matters/newsletters/issue13.htm, 2002.

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

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

[8] 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.

[9] Joost. http://www.joost.org/whatsjoost.html.

[10] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc of SIGCOMM*, Philadelphia, PA, Aug. 2005.

[11] T. Karagiannis, D. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proc of SIGCOMM*, Philadelphia, PA, Aug. 2005.

[12] 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.

[13] 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.

[14] Kontiki. http://www.kontiki.com.

[15] A. Kuzmanovic and E. Knightly. TCP-LP: Low-priority service via end-point congestion control. *IEEE/ACM Transactions on Networking*, 14(4), Aug. 2006.

[16] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit. Are file swapping networks cacheable? characterizing P2P traffic. In *Proceedings of WCW*, Boulder, CO, 2002.

[17] 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.

[18] A. Odlyzko. Internet pricing and the history of communications. *Computer Networks*, 36(5–6):493–517, Aug. 2001.

[19] P-Cube. P-Cube. http://www.p-cube.net/indexold.shtml.

[20] Packeteer. Packeteer PacketShaper. http://www.packeteer.com/products/packetshaper.

[21] Pando Networks, Inc. http://www.pandonetworks.com.

[22] M. Piatek, C. Dixon, A. Krishnamurthy, and T. Anderson. Liveswarms: Adapting bittorrent for end host multicast. Technical Report UW-CSE-06-11-01, University of Washington, 2006.

[23] D. Qiu and R. Srikant. Modeling and performance analysis of Bittorrent-like peer-to-peer networks. In *Proc. of SIGCOMM*, Portland, OR, Aug. 2004.

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

[25] B. Raghavan and A. C. Snoeren. A system for authenticated policy-compliant routing. In *Proc. of SIGCOMM*, Portland, OR, Aug. 2004.

[26] RouteScience Technologies, Inc. Route optimization for ebusiness applications. White Paper. Available at http://www.routescience.com, June 2003.

[27] O. Saleh and M. Hefeeda. Modeling and caching of peer-to-peer traffic. Technical Report TR 2006-11, Simon Fraser University, May 2006.

[28] Sandvine. Intelligent broadband network management. http://www.sandvine.com.

[29] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. In *Proc of OSDI*, Boston, MA, Dec. 2002.

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

[31] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of P2P traffic using application signatures. In *Proceedings of WWW13*, New York, NY, May 2004.

[32] 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.

[33] Statelog White Paper. Peer-to-peer and bandwidth management. http://www.staselog.com/whitepaper-p2p.pdf, Mar. 2004.

[34] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A mechanism for background transfers. In *Proc of OSDI*, Boston, MA, Dec. 2002.

[35] H. Wang, H. Xie, L. Qiu, A. Silberschatz, and Y. R. Yang. Optimal ISP subscription for Internet multihoming: Algorithm design and implication analysis. In *Proceedings of IEEE INFOCOM '05*, Miami, FL, Apr. 2005.

[36] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak. Cache replacement policies revisited. In *Proc of GP2P*, Chicago, IL, Apr. 2004.

[37] R. Woundy and J. Livingood. Comcast field tests. IETF P2P infrastructure workshop, May 2008.