Peer-Assisted Content Distribution with Prices

Christina Aperjis
Stanford University

Michael J. Freedman
Princeton University

Ramesh Johari
Stanford University

ABSTRACT

Peer-assisted content distribution matches user demand for content with available supply at other peers in the network. Inspired by this supply-and-demand interpretation of the nature of content sharing, we employ *price theory* to study peer-assisted content distribution. The market-clearing prices are those which align supply and demand, and the system is studied through the characterization of price equilibria. We discuss the efficiency and robustness gains of price-based multilateral exchange, and show that simply maintaining a single price per peer (even across multiple files) suffices to achieve these benefits.

Our main contribution is a system design—*PACE (Price-Assisted Content Exchange)*—that effectively and practically realizes multilateral exchange. Its centerpiece is a market-based mechanism for exchanging currency for desired content, with a single, decentralized price per peer. Honest users are completely shielded from any notion of prices, budgeting, allocation, or other market issues, yet strategic or malicious clients cannot unduly damage the system's efficient operation. Our design encourages sharing of desirable content and network-friendly resource utilization.

Bilateral barter-based systems such as BitTorrent have been attractive in large part because of their simplicity. Our research takes a significant step in understanding the efficiency and robustness gains possible with multilateral exchange.

1. INTRODUCTION

Peer-to-peer systems have been wildly successful as a disruptive technology for content distribution. Varying accounts place peer-to-peer (P2P) traffic as comprising anywhere between 35% and 90% of "all" Internet traffic, with BitTorrent accounting for its large majority [5]. Perhaps BitTorrent's biggest technical contribution is its content-exchange mechanisms—rate-based tit-for-tat [8], or *bilateral barter*—that users widely view as incentivizing uploads.

While BitTorrent's usage numbers are certainly impressive, there are some fundamental problems with its resource allocation and incentive mechanisms, even beyond potential "free-riding" attacks already observed [15, 19, 26, 24]. Namely, the system can only perform bilateral barter by matching up well-suited pairs of nodes that have disjoint subsets of a file (or, more generally, files). Yet the discovery of stable peering relationships is slow in practice—measured in the tens of minutes [7] or, at least for high-bandwidth peers, requiring a linear brute-force search of other participants to find similar reciprocation rates [24]—if such reciprocation exists at all. In the end, altruistic uploading often turns out to be critical for providing continued content availability [12].

From an economics perspective, many of these problems ultimately have to do with "market failure" in the system: it's hard to find good reciprocation with bilateral barter alone. But economics also offers an alternative—*market-based multilateral exchange*—where the system matches user demand for content to available supply at other peers in the network. As we discuss in Section 2, bilateral barter-based systems such as BitTorrent *implicitly* encode prices through their rate reciprocation mechanisms; by contrast, we explore the efficiency benefits of *explicit* price signals. Explicit prices can help identify which files are "most useful" to disseminate, where resource congestion is occurring, and who is providing useful content to the system. We briefly describe the significant efficiency gains possible through the use of a price-based multilateral exchange (see [2] for a full discussion).

Design of a price-based exchange is subtle: what exactly should we price? In Section 3, we consider a range of models, and conclude that simply maintaining a single price per peer suffices to achieve the benefits of price-based multilateral exchange. We also demonstrate the impact of currency on system dynamics. This analysis motivates our desire to use one price per peer in our system design.

Our main contribution is *a system design—PACE (Price-Assisted Content Exchange)—that effectively and practically realizes multilateral exchange*. Its centerpiece is a market-based mechanism for exchanging currency for desired content, with a single price per peer as advocated by our theoretical analysis. The system fully specifies the algorithmic buy-and-sell behavior of users' clients: Honest users are completely shielded from any notion of prices, budgeting, allocation, or other market issues, yet strategic or malicious clients cannot unduly damage the system's efficient operation. Still, users are incentivized to contribute resources, as each user's performance is affected by his (hidden) budget.

PACE has several desirable features. *First*, price management is straightforward. Price updating is fully decentralized

ACM CoNEXT 2008, December 9-12, 2008, Madrid, SPAIN

to the user clients, and price information is easily discoverable via existing architectures similar to BitTorrent trackers or even DHTs. We also show that our decision to use one price per peer leads to a simple price update implementation, as well as a simple and robust service discipline at uploaders.

Second, there are several incentive benefits of our design. For example, the exercise of "market power" by uploaders is transient, because the dissemination of a file immediately creates competitors. Further, the use of currency provides an incentive for users to remain available after downloading, as credit obtained can be spent later on other content.

Third, our model enables ISPs to easily incorporate signals for "network friendliness", and allows a dynamic adaptation to resource constraints across multiple domains. Our hierarchical network model captures the congestion points in a network for resource pricing, yet also supports the ability for ISPs to express preferences for long-haul traffic carriage. This model ensures that P2P traffic has a strong incentive to remain local when possible, or at least traverse wide-area links that yield more efficient network usage.

Finally, although PACE uses explicit prices, it is flexible enough to be adopted in a wide range of settings. For example, a content distributor can employ our system design on a trusted platform (*e.g.*, set-top boxes), in which case managing currency is vastly simplified. At the other extreme, decentralized currency mechanisms such as Karma [29], one-hop reputations [25], and BarterCast [28] may be used to support a system implementation that does not rely on central money management. In the middle lie systems that employ (potentially many) logically centralized banks, such as Dandelion [27] or an alternate design we sketch in §7.

The paper is organized as follows. Section 2 compares bilateral and multilateral exchange, and §3 compares various pricing schemes. Section 4 discusses the user incentives provided by our exchange model, while §5 presents our network model. Section 6 details the algorithmic mechanisms at the heart of our buy and sell clients, while §7 describes the services and protocols that complete the PACE system. Section 8 evaluates the resulting system in simulation, §9 discusses related work, and §10 concludes.

2. BILATERAL AND MULTILATERAL EXCHANGE

This section compares P2P system designs with bilateral barter, such as BitTorrent, and a market-based multilateral exchange of content enabled by a price mechanism to match supply and demand.

BitTorrent and its variants promote *bilateral* exchange between peers: a peer i uploads to a peer j if and only if peer j uploads to peer i in return. Of course, such an exchange is only possible if each peer has something the other wants. While such protocols are traditionally studied solely through the rates that peers obtain, in this section we provide an interpretation of these protocols through *exchange ratios*.

Let r_{ij} denote the rate sent from peer i to peer j in an instantiation of a BitTorrent swarm. We define the *exchange ratio* between peer i and peer j as the ratio $\gamma_{ij} = r_{ji}/r_{ij}$; this is the download rate received by i from j, per unit of rate uploaded to j. By definition, $\gamma_{ij} = 1/\gamma_{ji}$. Clearly, a rational peer i would prefer to download from peers with which he has higher exchange ratios, since that would increase his total download rate.

The exchange ratio has a natural interpretation in terms of prices. An equivalent story emerges if we assume that peers charge each other for content in a common monetary unit, but that all transactions are *settlement-free*, *i.e.*, no money ever changes hands. In this case, if peer i charged peer j a price p_{ij} per unit rate, the exchange of content between peers i and j must satisfy $p_{ij}r_{ij} = p_{ji}r_{ji}$, and the exchange ratio is $\gamma_{ij} = p_{ij}/p_{ji}$ (as long as the prices and rates are nonzero).

What is the exchange ratio for BitTorrent? A peer splits its upload capacity equally among those peers in its active set from which it gets the highest download rates. Let α be the size of the active set. Suppose all rates r_{kj} that peer j receives from peers $k \neq i$ are fixed and let R_j^α be the α-th highest rate that j receives. Let B_j be the upload capacity of peer j. Then, r_{ji} depends on r_{ij}. In particular,

$$r_{ji} = \begin{cases} B_j/\alpha & \text{if } r_{ij} > R_j^\alpha; \\ 0 & \text{otherwise.} \end{cases}$$

Thus for BitTorrent, the exchange ratio is $\gamma_{ij} = B_j/(\alpha \cdot r_{ij})$ if peer i is in the active set. We note that $\gamma_{i_1,j}$ and $\gamma_{i_2,j}$ may be different for two peers i_1 and i_2 in j's active set.

The exchange ratio γ_{ij} *decreases* with r_{ij} as long as peer i is in peer j's active set. Hence, a strategic peer i would prefer to choose r_{ij} as small as possible while remaining in j's active set. This behavior is exactly the approach taken by the BitTyrant [24] variation on BitTorrent. In fact, if all peers that upload to j follow this policy, then $r_{ij} = R_j^\alpha$ for all peers i in j's active set. Note that in this case, $\gamma_{ij} = B_j/(\alpha \cdot R_i^\alpha)$. Thus, peer j has the same exchange ratio to all peers i with which he bilaterally exchanges content.

Bilateral exchange is rather restricted. A peer is implicitly required to download only from those peers to which he uploads. Another option is multilateral exchange: let a peer accrue capital by uploading, and allow him to spend it however he wishes for downloading. This is achieved through prices on peers or files (or both) as explained in Section 3, where multilateral exchange is formally defined. In such a setting, some peer i may be downloading from peer j, even though i is not uploading to j, but to some other peer k.

In [2], we exploit the relationship between exchange ratios and prices to theoretically compare the two exchanges through their equilibrium allocations, where peers explicitly react to given exchange ratios or prices. We show that when exchange is restricted to being bilateral, then equilibria may fail to exist, may be inefficient if they do exist, and may fail to be robust to collusive behavior even if they are efficient. With multilateral exchange on the other hand, equilibria exist under general conditions, are efficient, and are robust to collusive behavior. Indeed, we show that a bilateral exchange equilibrium is also a multilateral exchange equilibrium exactly if an additional collusion-resistance property is satisfied (see [3] for full details). These results demonstrate that multilateral exchange is efficient and robust to strategic devi-

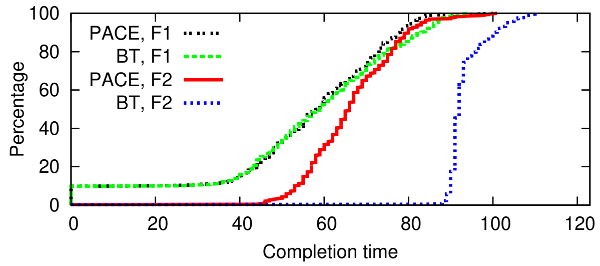

Figure 1: **Completion times for PACE and BitTorrent exchanges. File F1 started at 10% of the nodes; file F2 at a single node.**

Peer Optimization:

maximize $V_i(\mathsf{d}_i)$

subject to $d_{if} = \sum_j r_{jif}$ for all $f \in T_i$

$r_{ijf} = 0$ if $f \notin F_i$

$\sum_{j,f} r_{ijf} \le B_i$

Budget Constraint

$\mathsf{r} \ge 0.$

Budget Constraints:

PP $\sum_{j,f} p_j r_{jif} \le p_i \sum_{j,f} r_{ijf}$

PF $\sum_{j,f} p_f r_{jif} \le \sum_{j,f} p_f r_{ijf}$

PFP $\sum_{j,f} p_{jf} r_{jif} \le \sum_{j,f} p_{if} r_{ijf}$

Figure 2: **Optimization problem for price-based exchange.**

ations of cliques, while bilateral exchange is not in general.

Figure 1 provides a comparison of dynamic behavior. The figure shows the completion times of peers under multilateral exchange (PACE) and rate-based tit-for-tat (BitTorrent). We see that for files with constrained availability, peers complete significantly faster in multilateral exchange, demonstrating that it behaves well in dynamic settings. Section §8 describes our simulations in greater depth.

Given these benefits, we choose to use multilateral exchange for the system design. In the following section, we discuss the benefits of having explicit prices and currency in the system, and we provide a comparative study of a range of price-based multilateral exchange mechanisms.

3. PRICES IN PEER-TO-PEER SYSTEMS

As discussed in the previous section, multilateral exchange satisfies a number of desirable properties that bilateral exchange does not. In this section, we discuss how these benefits can be enabled. In §3.1, we compare a range of pricing schemes for multilateral exchange and conclude that simply maintaining a single price per peer suffices to achieve the benefits of price-based multilateral exchange. This result means that a more complex approach of variable pricing (at each peer) for different files or different chunks of files is, in fact, neither necessary nor beneficial. Intuitively, even with a single price per peer, variable prices for files still arise *across the network* since different peers supply different files. In §3.2, we consider how explicit prices, pricing per peer, and currency benefit system dynamics. We conclude by discussing the implications for designing a system for multilateral exchange.

3.1 Comparing Pricing Schemes

This section compares three pricing schemes for multilateral exchange: (1) one price per peer (denoted PP); (2) one price per file (denoted PF); and (3) one price per file per peer (denoted PFP). We compare the schemes through their equilibria. First, we show that all three are equivalent when transfers are only constrained by peer upload capacity. However, we then demonstrate that PF may be strictly worse than PP if the network topology is non-trivial. Finally, we show that PP and PFP yield equivalent equilibria, even when the network topology is non-trivial. Since explicitly pricing every file of every peer is much more complicated than only maintaining a single price per peer, our analysis suggests PP is the most desirable scheme.

In the formal model we consider, a set of peers N shares a set of files F. Peer i has a subset of the files $F_i \subseteq F$, and is interested in downloading files in $T_i \subseteq F - F_i$. Let r_{ijf} be the rate at which user i uploads file f to user j, and $d_{if} \equiv \sum_j r_{jif}$ be the total rate at which user i downloads file f. We use sans serif to denote vectors, *e.g.*, $\mathsf{d}_i = (d_{if}, f \in T_i)$ is the vector of download rates for user i. We measure the desirability of a download vector to peer i by a *utility function* $V_i(\mathsf{d}_i)$ that is nondecreasing in every d_{if} for $f \in T_i$. The upload capacity of peer i is denoted by B_i. We allow peers to bilaterally exchange content over multiple files, even though this is not typically supported by swarming systems like BitTorrent.

Figure 2 gives the peer optimization problem in price-based exchange. The first three constraints (giving download rates, ensuring peers only upload files they possess, and meeting the bandwidth constraint) are identical for all pricing schemes. The budget constraint ensures that the capital peer i spends for downloading does not exceed the capital the peer accrues by uploading. Given prices $(p_i, i \in N)$, $(p_f, f \in F)$ and $(p_{if}, i \in N, f \in F)$ for the PP, PF and PFP pricing schemes respectively, the corresponding budget constraints are shown in Figure 2.

An *equilibrium* is a combination of a rate allocation vector and a price vector such that all peers have solved their corresponding optimization problems. In this case, the prices have exactly aligned supply and demand: for any i, j, f, the transfer rate r_{ijf} is simultaneously an optimal choice for both the uploader i and downloader j. We next give the equilibrium definition for the PP pricing scheme. (Similarly, an equilibrium can be defined for the two other pricing schemes.)

Definition 1 *The rate allocation* $\mathsf{r}^* = (r_{ijf}, i, j \in N, f \in F)$ *and the peer prices* $(p_i^*, i \in N)$ *with* $p_i^* > 0$ *for all* $i \in N$ *constitute a* ***PP equilibrium*** *if for each peer* j, r^* *solves the Peer Optimization problem given prices* $(p_i^*, i \in N)$.

We first observe that if transfers are only constrained by the upload capacity of peers, then the PF and PP schemes are equivalent in terms of equilibria, *i.e.*, an equilibrium for one scheme exists if and only if there exists an equilibrium with the same rate allocation for the other pricing scheme. In particular, given PF equilibrium prices $(p_f^*, f \in F)$, peer i will only be uploading files in $\arg\max_{f \in F_i}\{p_f^*\}$. Setting $p_i^* = \max_{f \in F_i}\{p_f^*\}$, we get an equilibrium for the PP scheme, since the optimization problem of each peer does not change. Similarly, if $(p_i^*, i \in F)$ are PP equilibrium prices, then $p_f^* = \min_{i: f \in F_i}\{p_i^*\}, f \in F$ are PF equilibrium prices which yield the same rate allocation.

Given a non-trivial network topology, however, links other than peer access links may be congested. These links need to be priced as well to ensure efficient network usage. Again abusing notation, we denote the price of link ℓ by p_ℓ. For now we assume that we can price every link in the network, an assumption we waive in §5. When peer i is downloading from peer j, i pays j, but also all links that i's traffic traverses.

Network links are priced in order to make peers internalize their effect on the network, and not for profit. Thus, it is desirable to rebate any payments related to network costs to peers. We will assume that whatever is paid to traverse links in the network is rebated equally to all peers; however, our results also hold for other rebating schemes.

When the network topology is non-trivial, we can modify the budget constraints to include the payment to the network on the left hand side and the rebate from the network on the right hand side. An *equilibrium* now is a combination of a rate allocation vector and a price vector (which includes prices for links) such that all peers have solved their corresponding optimization problems and the total traffic that traverses each link does not exceed its capacity.

The following example shows that when the network is non-trivial, equilibria may fail to exist under the PF scheme, even though they exist for the PP and PFP schemes.

Example 1 *There are four peers and two files, with file allocation and demand as shown in Figure 3. The network has two clusters, consisting of peers* $\{1,2\}$ *and* $\{3,4\}$*, with a bidirectional link* ℓ *of capacity* 1 *connecting them. Peers* $\{1,3\}$ *have file* f *and want file* g*; peers* $\{2,4\}$ *have file* g *and want file* f*. The peers' upload capacities are* $B_1 = B_4 = 8$ *and* $B_2 = B_3 = 2$.

This system has no equilibrium under the PF scheme. If $p_\ell = 0$*, peers demand* $d_1 = (p_f/p_g)B_1$*,* $d_2 = (p_g/p_f)B_2$*,* $d_3 = (p_f/p_g)B_3$ *and* $d_4 = (p_g/p_f)B_4$ *when optimizing. The market clears only if* $d_1+d_3 = B_2+B_4$*, which implies* $p_f/p_g = 1$*. But then* $d_1 = 8$*, which is not feasible, since the maximum total rate at which peer 1 can download is 3. If* $p_\ell > 0$ *and there is a single price per file, then peers only download locally and, from the market clearing condition, we get* $p_f/p_g = 4$ *in one cluster and* $p_f/p_g = 1/4$ *in the other, which is a contradiction.*

On the other hand, under the PP scheme, there is an equilibrium with prices $(p_1, p_2, p_3, p_4, p_\ell) = (1,3,3,1,2)$ *and rates* $(d_1,d_2,d_3,d_4) = (3,7,7,3)$*. Peers 1 and 4 download at rate 2 locally and at rate 1 remotely from each other. The revenue collected from the link* ℓ *is rebated equally to all peers, which allows peer 2 to download more than* $(p_1/p_2)B_2$.

The preceding example shows that for general network topologies, the existence of a multilateral equilibrium for the PP scheme does not imply the existence of an equilibrium for the PF scheme. The reason is that a file may be uploaded at different prices at different parts of the network.

PFP is the most general pricing scheme. The following result shows that PP is equivalent to PFP in terms of equilibria. The proof (which can be found in [3]) is similar to the proof

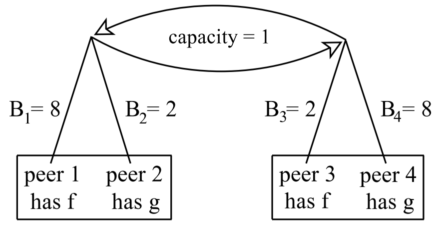

Figure 3: **System with peers $\{1,2,3,4\}$ and files $\{f,g\}$. Peers are located in two clusters; transfers are constrained by bandwidth constraints of peers and the inter-cluster link.**

of equivalence between PP and PF in the trivial network setting. It shows that given equilibrium prices for one pricing scheme we can construct prices for the other pricing scheme that yield the same equilibrium allocation.

Proposition 2 *For any network topology, there exists a multilateral equilibrium for the PP scheme if and only if there exists a multilateral equilibrium for the PFP scheme.*

We conclude that one price per peer is sufficient to identify heterogeneity in the system. Intuitively, the upload capacity of a peer's access link is the local resource that becomes congested, and in market design it is typically the case that one price is required for each congestible resource; hence one price per peer suffices for multilateral equilibrium. The argument holds not only for different files, but also for different chunks of the same file. In light of Example 1, a price for each peer is the minimal amount of information needed.

The PP scheme provides many practical benefits as well. It greatly reduces the number of prices that need to be maintained, compared to PFP pricing. Further, it simplifies price discovery and leads to a natural service discipline for uploading files, as discussed in §3.2.

3.2 Dynamics

The preceding section considers equilibria, *i.e.*, a *static* setting. However, since it is hard to know a system's equilibrium prices or allocation in advance, we need to consider several issues related to *dynamics*: how downloaders and uploaders are matched (peer discovery), how out-of-equilibrium prices are updated (price discovery), and which requests uploaders satisfy (service discipline). In this section we discuss the role of *explicit* per-peer pricing in aiding dynamics; we also briefly discuss the advantages of explicit currency.

A significant advantage of *explicit* per-peer prices is that they enable fast peer discovery. This short discovery time significantly improves on that needed by systems with *implicit* prices, *e.g.*, such as BitTorrent's rate-based exchange ratios, which have long discovery times. These implicitly-priced systems need to perform a brute-force search across their peers; this has been found in practice to sometimes take tens of minutes [7], and, at least for high-bandwidth peers, requires an asymptotically-linear search to find similar reciprocation rates.

For price discovery, a simple mechanism is to update the prices of peers and links according to the corresponding *excess demand*. In particular, a price should increase if demand exceeds supply, and decrease if demand trails. But how to define supply and demand for a peer? If the PP scheme is employed, a peer's observed demand is the sum of all received rate requests, and his supply is equal to the upload capacity of his access link (within a fixed time period). If the PFP scheme is used, excess demand is more complex, since it needs to be calculated for each file separately, and a peer's supply for a file depends on the prices (it is optimal for each peer to only upload his most expensive file). Thus the PP scheme leads to simpler price dynamics. The benefits of the PP pricing scheme are even more apparent for unpopular files for which requests are relatively rare.

PP pricing leads to a natural service discipline for uploading files, well-aligned with a peer's incentives: serve requests sequentially and without preemption. The service discipline for PFP pricing is less clear, however: Serving requests only for the highest-priced file may not fully utilize a peer's available resources, while serving requests sequentially is not profit maximizing.

Allowing peers to store and exchange currency over time also has significant benefits for a system in a dynamic setting. First, peers can engage in trade *before* equilibrium prices are reached. Second, peers can reach a rate allocation by *trading in a decentralized fashion*, without requiring a central authority to clear the market by matching uploaders and downloaders. However, incorporating currency into a system introduces its own complications, since the user experience could potentially be complicated, and peer exchanges and credit balances need to be secured. We demonstrate how these issues can be handled in §6 and §7, respectively.

3.3 Implications for System Design

In the previous sections, we argued that explicit prices and currency facilitate multilateral exchange and showed that one price per peer suffices to achieve the benefits of multilateral exchange. Our system design (PACE) will enable multilateral exchange through currency and the per-peer pricing scheme. The high-level picture is the following. Given prices of peers and network links, a peer requests download rates from other peers in the network. For downloading, a peer pays the uploading peer and the links its traffic traverses. Payments to links are rebated equally to all peers. A peer serves requests sequentially without preemption, and updates its price according to the mismatch between requests received and available capacity.

In §4, we argue that this high level design in general provides the right incentives to users. We then make the system design more practical in terms of network prices and end-user decisions. Since it is probably infeasible in practice to price all links in the network, we propose in §5 a hierarchical model that separately prices most of the bottlenecks that lead to supply constraints. In §6, we argue that pricing mechanisms, rather than being directly exposed to end-users, can serve as algorithmic devices to ensure efficient exchange, and simplify the user experience. Section 7 describes the services and protocols needed to complete the PACE system.

4. USER INCENTIVES

In this section we study the incentives provided to users in our multilateral exchange model. *First*, we note that our system encourages efficient use of resources in a *large* P2P system. If the system is large, users cannot accurately anticipate how their actions affect prices, *i.e.*, users have difficulty in predicting the evolution of demand, supply, and prices.

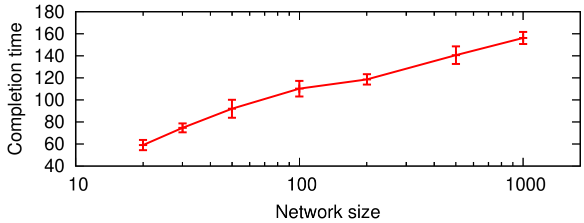

Figure 4: **Completion time grows $O(\log(|\mathbf{network}|))$. Each data point shows the average performance over five runs—error bars show standard deviation—with 10 peers per cluster.**

Second, users are given incentives to contribute. In particular, users are implicitly incentivized both to contribute a high percentage of their upload capacities and to share high-value content, since high-value files will typically increase a user's price. Moreover, with currency stored over time, users are incentivized to contribute even if they are not currently downloading. We also note that a user is paid for delivered rates, so a user does not profit by advertising a higher upload capacity than the one he has.

Third, network prices align user incentives with efficient network usage. If network links are priced correctly, we expect network prices to reflect congestion. Then users internalize their effect on the network, since they have to pay for all network links they use. For instance, among peers with the same price, a user prefers to download from the peer with the smallest total network cost, which we expect to correspond to the least congested route. On the other hand, sellers do not benefit from network-cost-related payments, and so the system does not create a perverse incentive for sellers to prefer remote transfers.

Finally, one potential concern in a market-based system is the phenomenon of *market power*: Users may try to manipulate prices higher than the laws of supply and demand dictate. This effect is mitigated significantly in a market with many sellers, where market manipulation cannot significantly increase profit [21]. To illustrate this, suppose k peers have a file desired by other users in the system. Let $d(p)$ be the demand for the file at price p, and B_j be the bandwidth of peer j. The equilibrium price p^* satisfies $d(p^*) = \sum_{j=1}^{k} B_j$. Peer i would benefit from increasing his price to $p_i > p^*$ only if

$$\frac{B_i}{d(p)} \geq -\frac{d(p)-d(p^*)}{p-p^*} \cdot \frac{p}{d(p)}.$$

The right-hand side represents the demand elasticity, *i.e.*, the percentage change in quantity demanded, divided by the percentage change in price. Thus, user i will not be able to exert market power as long as enough sellers compete to upload the file relative to the elasticity of demand.

Market power may still be an issue if only a few users have a file, yet any system can suffer if such users choose to dictate terms to the remainder. In our setting, such users are often the "seeders" of files and *want* to see their content

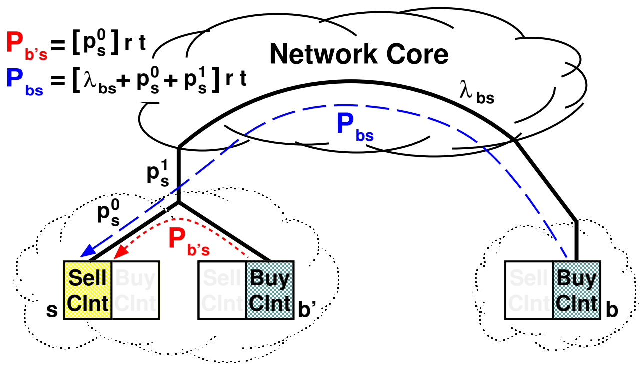

Figure 5: ***Network Model.*** **Clusters of well-connected peers connect across wide-area links. To download file f from sell client s, local buy client b' pays $P_{b's}$ and remote b pays P_{bs} to s.**

disseminated. Many peers in existing P2P systems exhibit such altruistic behavior.

More generally, sellers *create* other uploaders in the very act of uploading; thus, market power is at best a transient phenomenon, since other sellers quickly emerge as competitors. Further, the number of competitors grows exponentially in time: if a file chunk is always transferred from one user to another in one time period, then after t time periods, $O(2^t)$-times more users will have it. Indeed, this asymptotic behavior is supported by our simulation results; Figure 4 shows that the completion time of all nodes to download a file appears to grow logarithmically with network size, as expected.

5. HIERARCHICAL NETWORK MODEL

In §3.1, we associated a price with every link in the network in order to ensure efficient network usage. However, this is probably infeasible in practice, given a lack of network topology and routing information, inaccurate bandwidth capacity estimations, and computational complexity. Thus, we propose a hierarchical network model, outlined in Figure 5, that separately prices most of the bottlenecks that lead to supply constraints.

We observe that, to a first approximation, we can view the entire network as composed of *local clusters*, connected together by the wide-area core. Our model assumes that most bottlenecks—especially those due to transient congestion—are at access links, where we can accurately estimate capacity and adapt prices accordingly. Rare bottlenecks in the core are captured via slowly-changing *network prices*. Different network prices may be associated with different pairs of clusters. Section 7 discusses how these network prices can be set, potentially promoting cooperation between ISPs and P2P systems. While less expressive than a price-per-link, our design compares favorably to typical AS-level clustering approaches [16, 1], which statically restrict peers to their local neighbors and do not capture dynamic resource constraints.

Our system design does not specify precise capacity requirements for local clusters and the wide-area network. We anticipate that nodes in the same local cluster share relatively high-capacity links to each other (such as within a LAN or a switched university network) and that transmission across the wide area will involve shared, lower-capacity access links to the Internet (*e.g.*, a DSL connection or a university's external link).

To capture these constraints, a peer s maintains two prices, p_s^0 and p_s^1, corresponding to the immediate upstream link at s and the access link shared by all nodes in the same cluster as s, respectively. One could extend this hierarchical model to capture additional choke-points at the network's edge if needed and price these links separately and accordingly. In fact, different peers could maintain different numbers of prices, based on their local network conditions (and this would be largely transparent to buyers). For simplicity, we restrict our further consideration to these two prices.

If peer b' is within the same local cluster as s, b' pays $P_{b's} = p_s^0 \cdot r \cdot t$ to s for a transfer of duration t at rate r. Such a payment is shown by the red dotted arrow in Figure 5.

If peer b is not within the same local cluster, the traffic must also traverse the access link for s's cluster and the wide-area core. Peer b pays $P_{bs} = (\lambda_{bs} + p_s^0 + p_s^1) \cdot r \cdot t$ to download from s (the blue dashed arrow),[1] where λ_{bs} is the network price between the clusters of b and s. Of this, only $(p_s^0 + p_s^1) \cdot r \cdot t$ is paid to s; the remainder is rebated equally to all peers.

6. PACE CLIENT MECHANISMS

One typical complaint against explicit pricing and currency is that the process of setting prices and bidding for goods becomes a usability hurdle. Indeed, this hurdle is seen by the designers of systems such as MojoNation [30] as their major reason for failing to be widely adopted.

However, pricing mechanisms, rather than being directly exposed to end-users, can serve as *algorithmic devices* to ensure efficient exchange: We can expose a very simple interface to users and have users' software optimally compute their buy and sell behavior. We expect that even strategic users cannot gain any significant benefit from operating in a manner other than that specified by our algorithms (per §4).

This section describes the design and algorithmic mechanisms of buy and sell clients in PACE. Clients interact with each other across both local- and wide-area networks; when chunks of a file are downloaded by a buy client from a sell client, (virtual) currency flows in the opposite direction.

6.1 Sell Client

The sell client interface allows the user to declare (1) the files he is willing to upload and (2) the total upload capacity he is willing to commit. The sell client s maintains prices p_s^0 and p_s^1, corresponding to the two edge resources being priced (per §5). This section details the price update rules and service discipline used by the sell client.

The sell client s follows a very simple price update rule in principle. Prices increase if demand exceeds supply and fall if the opposite; we use a multiplicative increase/multiplicative

[1] Note that any remote peer b that downloads from s pays to traverse the access link for s's cluster, but not for its own access link. This asymmetry means that congestion on the downstream link from the wide-area core to b's cluster may not be correctly priced. This congestion applies to all remote files, however, and since local prices are typically lower than remote prices, buy clients will generally prefer local sources anyway.

decrease rule, so that convergence is insensitive to the units in which prices are measured. A difficulty arises because there are *two* scarce resources that are being priced; we now describe the price update schemes for both p_s^0 and p_s^1.

Sell client s estimates demand and supply over a fixed time interval. To update p_s^1, demand is estimated as the total requested download rate originating at *remote* buy clients, provided they offer at least p_s^1. A request is counted regardless of whether sufficient capacity existed to serve it.[2] Supply is estimated as the sell client's upload capacity on its access link, B_s^1. Estimating B_s^1 may be done by tracking the maximum aggregate throughput ever seen across all uploads. (Note that systems such as BitTorrent similarly use edge-capacity estimation, *e.g.*, to set its active set size [24].)

The approach to update p_s^0 is similar. Demand is estimated by aggregating the rates of all download requests from both local and remote buy clients, taking the access-link bandwidth constraint into consideration. In particular, demand is equal to the sum of local requests and the minimum of remote requests and B_s^1. Supply is the upload bandwidth on the sell client's immediate link, B_s^0.

Finally, we consider the sell client's service discipline. Incoming download requests are served in the order of arrival; they can be immediately notified of acceptance, subject to available capacity constraints. The fact that *per-peer* pricing gives the seller no real incentive to queue requests simplifies system design: If we queued requests, then buy clients would need to maintain a large number of outstanding requests, and reason about spending over a long time horizon.

6.2 Buy Client

The buy client's interface allows the user to choose (1) the files he is interested in (denoted by T_i for user i) and (2) a *savings rate*, *i.e.*, the percentage η of the user's current budget that should be saved for the future. During each fixed time interval, our buy client sets aside a fraction η of the user's bank-account balance, and divides the remainder equally among all files the user wishes to download. For each such file f, the buy client follows a simple algorithm: First, given a set of sell clients that have f, order these clients j by the increasing *total* price the buy client has to pay for downloading from j (including network prices and remote access prices, if applicable). Then, at each seller j in this order, the buy client spends as much of its budget committed to f as possible. If j's upload capacity is exhausted, the buy client moves to the next sell client in the list. The buy client stops when it exhausts its budget for f. Any unused budget is split evenly between files in $T_i - \{f\}$.

This algorithm, though quite simple, can be interpreted through a foundational utility model for the user. In particular, the buy client behaves *as if* a user's utility for downloading is $\sum_{f \in T_i} \log d_f$, where d_f is the total download rate obtained for file f. Let M be the user's current bank account balance and $\bar{p}_f$ be the average price he has to pay for file f over available sellers (in the order described above). Given that the user wants to reserve η percentage of his current budget, the client solves the following problem:

$$\begin{aligned} \text{maximize} \ & \sum_{f \in T_i} \log d_f \\ \text{subject to} \ & \sum_{f \in T_i} \bar{p}_f \cdot d_f \ \le \ (1-\eta)M \\ & d_f \ge 0, \ \forall f \in T_i \end{aligned}$$

The optimal solution recovers exactly the above algorithm: It is optimal to divide $(1-\eta)M$—the budget allocated to this period—equally among all files in T_i. (Note that although our interpretation is in terms of per file prices, our implementation is in terms of per peer prices.)

There is a tradeoff between simplifying the user's experience and allowing the user to personalize his utility function. The above mechanism favors simplicity over descriptive power; other approaches can prefer the opposite.[3]

7. PACE SYSTEM DESIGN

This section describes the services and protocols needed to complete the PACE system. We discuss how our system can be employed in both a distrustful setting (where peers may run their own buy and sell client implementations), and a trusted platform (where end-clients obey the protocol).

In particular, there are compelling settings where *end-clients may be trusted* to obey the protocol, such as with set-top boxes. Dedicated hardware/software platforms are increasingly popular—*e.g.*, Comcast On-Demand, Verizon FiOS TV, Microsoft Xbox Live, AppleTV, the Netflix/LG partnership, and so on—with peer-assisted delivery on the horizon. In these cases, PACE's explicit prices would still serve as a mechanism for efficient resource discovery and allocation, especially for multi-file download situations.

To enable users to act as both buyers and sellers in currency-backed content distribution, PACE should support the following functionality:

1. ***Resource discovery.*** A buy client should learn a set of low-cost sell clients from whom to download content.
2. ***Network friendliness.*** Service or network operators can express preferences for peer download behavior across the wide area; buy clients are incentivized to follow these preferences.
3. ***User management.*** The system concerns itself with user registration, bootstrapping, and collusion.
4. ***Transactions and currency security.*** Exchanges between two parties should be *ε-fair*: sell clients should receive payment within an ε of the content transmitted for the agreed price; and buy clients should only pay for what they receive. Clients should not be able

[2]In practice, demand might be somewhat overestimated, as a buy client can issue several sequential requests to different sell clients until a successful download. However, demand is overestimated only when there already is excess system-wide demand, so prices would have increased anyway.

[3]For instance, we could allow the user to assign a weight w_f for each file $f \in T_i$, and then maximize $\sum_{f \in T_i} w_f \log d_f$. In this case, a simple greedy algorithm which allocates budget to files according to these weights is optimal.

to forge system currency nor spend more money than their current balance allows.

We describe our design for achieving these four properties, organized as above. Resource discovery and network friendliness apply to both trusted and distrustful settings; user management and security are relevant only for the latter.

System overview. At a high level, the PACE architecture is composed of four main components. First, users run both *buy* and *sell clients* to trade content for virtual currency. Sell clients advertise their existence (liveness), shared files, and current price to a *rendezvous service*, where buy clients also connect to discover nearby instances of sell clients offering desired content. A *network price service* specifies the network prices to accommodate network operator preferences.

In a non-trusted platform, a *currency mechanism* is also needed to securely track each user's accrued capital. Both decentralized currency mechanisms (*e.g.*, Karma [29], BarterCast [28], one hop reputations [25]), and logically centralized banks (*e.g.*, Dandelion [27]) may provide sufficient security in a non-trusted platform. For concreteness we fully specify a centralized currency mechanism that secures transactions and currency in §7.4.

We note that while some *logical* centralization may be considered undesirable, it is common to many of today's deployed peer-to-peer systems, from Gnutella through Tor, Skype and BitTorrent. These logically-centralized services can be made scalable: the PirateBay BitTorrent site was, as of January 2008, tracking over one million files and 10 million peers [23]. That said, some system services may be federated between existing network providers.

We envision multiple rendezvous, network price services, and—in the distrustful setting—banks to exist simultaneously, run by providers seeking to disseminate *collections* of files, which may range from large multimedia libraries (*e.g.*, iTunes and YouTube), to aggregates of user-hosted files (*e.g.*, the PirateBay BitTorrent tracker), to a corpus spanning the entire web (*e.g.*, CoralCDN [10]). We logically separate these services because different ecosystems may arise around all three: rendezvous is file-specific; banks are currency-specific; and yet network prices are independent of files and currency.

7.1 Resource and Price Discovery

There are a number of feasible engineering designs that PACE implementations could use for peer discovery, from logically-centralized trackers (à la BitTorrent), to a federated application-level anycast service (*e.g.*, OASIS [11]), to distributed hash tables indexing clients' addresses [10]. Because each user only needs to publish a single price (or one per hierarchical access link, given our network model), prices can be stored within the rendezvous service without significant communication overhead for updating. Thus, a query to the rendezvous service—which takes a file identifier and buy client information as input—can immediately return a randomized set of IP addresses with minimal published cost, potentially biased by network cluster locality or inflated by additional network prices, which we discuss next.

Upon discovering a set of sell clients with chunks of interest, the buy client determines their latest prices, allocates its budget across desired files, and prioritizes transfers which cost less, per §6. If two or more prices are identical, buy clients prefer those with smaller network prices.

7.2 Network Friendliness

Beyond leveraging users' prices to minimize resource congestion, PACE benefits network operators by having clients avoid expensive network links whenever possible. For a transfer traversing the wide-area Internet, the *network price* λ is added to the sell client's file price p_s, per §5. Thus, network prices between the clients' clusters need to be discovered.

PACE implementations and deployments have various design choices here as well. Well-suited for immediate deployment, a third-party service could independently determine network prices via network measurements [11, 20] or explicit ISP input. Alternatively, a buy client's network provider could run a lookup service itself (as with DNS and DHCP) and use these network prices to perform *policy-based traffic engineering* on its intra-ISP and egress traffic, taking its AS peering and transit relationships into account. In the longer term, such network prices may be propagated *between* ASes, perhaps alongside BGP announcements, and thus enable ISPs to better express their preferences for P2P data transfers. This inter-ISP final proposal introduces interesting incentive questions when coupled with current billing practices, so we leave this to future work.

In the preceding section, we proposed that the rendezvous service could take sell clients' prices into account when determining which clients to return. There are various design choices to be made, however, when incorporating network prices into this model. For a loose coupling between the two services, the rendezvous service may only incorporate peers' file prices or coarse-grained clustering information in its selection—the latter akin to coarse-grained locality proposals for peer-assisted CDNs [16, 1]—after which buy clients would subsequently lookup network prices themselves. This is well-suited for federated deployments in which ISPs provide network price services. On the other hand, if rendezvous and network price services are more tightly coupled, the exact network price could be considered in the selection criteria, leading to greater efficiency.

7.3 Managing Users

This section discusses several issues on managing user identities which are important when *strategic* users are involved. Until now, we have assumed that clients do not collude with one another, either as cliques of real-world users or as part of a Sybil attack [9]. While we can cite the traditional techniques to limit the scale of such attacks—such as analyzing "introduction" networks (*e.g.*, via SybilGuard [32]), leveraging real-world scarce resources (*e.g.*, phone numbers as in Google's GMail), or even requiring membership subscription via real-world money—the limited benefit of collusion in our setting bears some additional comments.

In PACE, a clique of clients cannot increase its aggregate wealth by combining trade amongst themselves with service to non-colluding clients at market rates; this is a consequence of the fact that multilateral exchange is robust to collusive behavior, as discussed in §2. This differs greatly from many reputation systems or systems based on download/upload rates, where shilling reputation announcements or faking content transfers [18], respectively, can increase a clique's aggregate balance.

The system must have a method to bootstrap new users. If Sybil attacks are less of a concern, new users can be granted money upon joining the system as a bootstrapping method. Alternatively, a new user can be offered the ability to download content for free from the operator: The redistribution of this in-demand content—not necessarily desired by the new user itself—can enable the system to provide better quality-of-service for its published content, while allowing users to earn initial capital by contributing upstream resources.

While these "free money" approaches may be susceptible to strategic Sybils—unless Sybil attacks are adequately protected by the registration process—-these same problems exist when bootstrapping most other incentive schemes of which we are aware, even if these systems do not explicitly use currency. Indeed, BitTorrent's performance reliance on seeders and its price-discovery mechanism (optimistic unchoking) are precisely "free money" and have been attacked as such [19, 26]. But because PACE maintains currency as stored value, this bootstrapping phase only occurs when *new users* join the system, not for *every file* being distributed and *every exchange* being transacted.

7.4 Securing Transactions and Currency

While the problem of securing transactions and currency may not arise when the system is deployed on a trusted platform, they must be considered in mutually-distrustful environments where peers may run their own buy and sell client implementations. While decentralized currency mechanisms, such as Karma [29] and BarterCast [28], may be applicable, this section discusses how security can be achieved with a *logically* centralized bank.

Securing Transactions. Exchanges between buy and sell clients should be ε-*fair* in that the goods they ultimately exchange are equal (within some small ε factor) with respect to the agreed-upon payment P. That is, if one party fails during the transaction after only $\frac{i}{k}$th of the content is transferred, and $\varepsilon = 1/k$, then the buy client should pay $\frac{i-1}{k}P \leq P' \leq \frac{i}{k}P$. We next describe how this property can be achieved.

To initiate a chunk transfer, sell client s sends a counter c_t and the agreed-upon price p to buy client b, who generates a commitment to s:

$$\sigma \leftarrow \mathsf{sign}_b(b,\ s,\ P,\ \Lambda,\ k,\ h_0,\ \hat{h},\ c_t)$$

where sign is a cryptographic signature with message recovery in the "hash-then-sign" paradigm [6]. $P = p \cdot r \cdot t = p \cdot |chunk|$ is the full payment for the chunk, of which Λ is the network cost. This payment P can be split across k micropayments. h_0 and $\hat{h}$ are outputs of a cryptographic hash function, generated as follows. The buy client first selects some random string h_k, then recursively computes a hash chain: $h_{i-1} \leftarrow \mathsf{hash}(h_i)$ for $i = k \ldots 1$. The buy client also creates a "shortcut" $\hat{h} \leftarrow \mathsf{hash}(\mathsf{S}, h_k)$ for global constant S.

At the beginning of a transfer, the buy client sends σ and its certified public key to s. Then s verifies both b's public key and σ's contents and signature. The sell client continues to transmit only if it receives a successive pre-image of the hash chain $h_i = \mathsf{hash}^{-1}(h_{i-1})$ every $\frac{1}{k}$-th of the transfer.

To deposit a payment at a bank (discussed next), b provides the tuple $\langle h'_i, i, \sigma \rangle$. The bank verifies σ and that the statement is not being replayed (using the counter c_t, also discussed next). If these checks succeed, the bank determines whether $i = k$, in which case the bank checks $\hat{h} \stackrel{?}{=} \mathsf{hash}(\mathsf{S}, h'_i)$. If $0 < i < k$, the bank verifies $h_0 \stackrel{?}{=} \mathsf{hash}^i(h'_i)$, *i.e.*, recursively applies the hash function i times. If this check passes, the bank executes the transaction for an amount $P' = \frac{i}{k}P$; that is, it debits b with the amount P', credits s with $P' - \frac{i}{k}\Lambda$, and reserves the (potentially zero) payment $\frac{i}{k}\Lambda$ for network costs.

Thus, we see that the bank's computational overhead is one "double-entry book-keeping" transaction, one signature verification, and one cryptographic hash in the normal case. In the case of truncated transfers, this overhead increases to at most k hashes, instead of one. (This use of $\hat{h}$ is a pure computational optimization.)

To further reduce the load at a bank, sell clients can aggregate multiple payments from the same buy client before depositing them as one transaction. Repetitive behavior is especially likely once two well-located peers discover one another and transfer many chunks, *e.g.*, those peers belonging to the same network cluster.

Securing Currency and Balances. To broker the exchange of files belonging to some *collection*, a content provider both runs a bank which manages currency used by the collection, and acts as a user registration authority for peers seeking to download files from the collection. The set of files that belong to a collection can be dynamic; banking/registration providers could thus offer their services to multiple publishers of collections.

Users are identified by public/private key pairs. When registering a new user, the bank issues a signed certificate to the user (with a short expiry time) attesting to its membership. This certificate is used by sell clients to verify the membership of buy clients without online checks. Banks must therefore refresh the certificate of active clients every expiry period; these updates can be piggy-backed on other control traffic, *e.g.*, deposits.

For each registered user, the bank stores, at a minimum, a hash of its public key, its current balance, and a monotonically increasing counter c_{last} of its last deposit. To prevent sell clients from replaying deposits, the counter c_t included in a deposited payment must be strictly greater than the buy client's c_{last}. If the deposit succeeds, c_{last} is set to c_t.

On the other hand, the bank must also prevent buy clients from issuing payments for money they do not have. For increased scalability, we have suggested a model by which individual sell clients can aggregate payments over short time horizons into single deposits. Of course, there then exists some window of vulnerability during which buy clients may overdraw on their accounts. Thus, if the bank ever detects that a client maintains a negative balance in its account, it should suitably punish or evict that client—by not renewing its membership certificate—from its network.

One could, of course, reduce this potential fraud by minimizing both sell clients' aggregation of payments and buy clients' certificate expiry times. In the limit, sell clients could perform an *online* check at the start of a transfer (escrowing P for some duration), and avoid any potential for overspending, at the cost of greater bank load and client latency.

We believe that these logically-centralized banks can be easily scaled via traditional replication and partitioning strategies; indeed, the systems community has several decades of experience building highly-scalable and available transaction-processing systems. Further, given that currency is only virtual, one may be able to slightly weaken consistency or freshness guarantees. Finally, as PACE's network model specifically encodes the hierarchical network structure for ISP-friendliness, network providers may use (currency-agnostic) local banks to aggregate local transactions to reduce load, especially given the extent to which transactions in PACE are executed locally. This layer of federation adds additional complexity, so we leave its full consideration to future work.

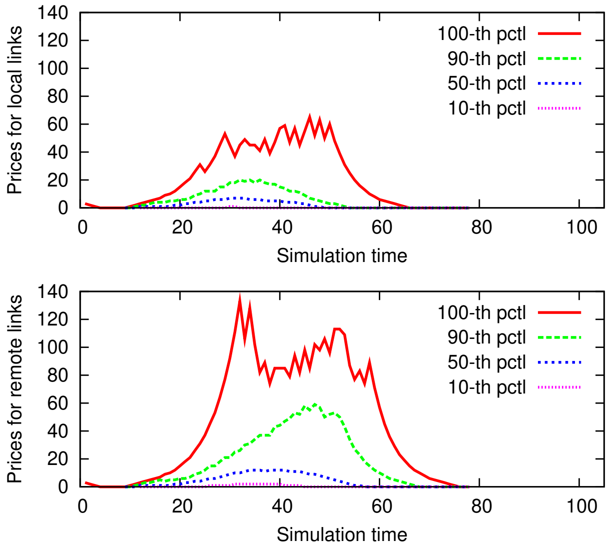

Figure 6: **Local links (top) are cheaper than remote (bottom).**

8. SIMULATION ANALYSIS

In this section, we evaluate the following hypotheses through simulation. (1) File prices reflect resource constraints. (2) Contributing upstream capacity improves a user's performance (and thus incentivizes such behavior). (3) File prices yield efficient dissemination across multiple files. (4) Efficient network utilization is enabled (compared to BirTorrent).

Simulator design and configuration. We model network connectivity and capacities using a hierarchical topology generated by BRITE [22], with clusters of nodes connected by AS-level links. In the following experiments, capacity between local hosts is 100 units/round, while that across wide-area links follows a heavy-tailed distribution on $(10, 1024)$. Each file is comprised of 50 chunks, each of fixed size (25 units). We use these generated graphs to compute end-to-end network capacities; for simplicity, however, we model all pairs of nodes as having independent links, *i.e.*, we do not model cross-traffic congestion. We also assume simple fixed transmission rates (*e.g.*, no TCP slow start). Network prices are static and computed as an inverse logarithmic function of capacity, ranging $[0, 5)$. Money collected from network prices are rebated equally to all peers.

The 8,000-line Python simulator operates in synchronous time steps, with PACE buy and sell clients behaving as specified in §6. Our BitTorrent-like implementation, used for some comparisons with PACE (*e.g.*, in Fig.1), captures the main aspects of the BitTorrent protocol: optimistic unchoking (with 2 slots), active set sizing (with $max(4, \sim\sqrt{B^1})$ slots), and rate-based tit-for-tat with equal-split bandwidth allocation (all per [24]). Both protocols use a local-rarest-first policy for chunk selection, and all peers have a neighborhood set of at most 80 peers.

We first evaluate how the system handles a flash-crowd for a single file: All users are interested in the same file, initially published at a single, random sell client (with an initial price of 1). In most experiments, peers' arrivals take a Poisson distribution with a mean of 20 time units (rounds); after finishing a download, peers stop sharing a file after some time (which is exponentially distributed, also with a mean of 20 time units).

The following results are for a network of 500 peers, comprised of 50 clusters of 10 nodes each. All nodes begin with 2,000 currency units. A randomly-chosen 50% of nodes are "freeloaders," *i.e.*, they never upload content. Non-freeloaders upload—and thus accrue capital—even after finishing their file download(s). We later consider the multiple file case.

The system behaves similarly when nodes' initial currency varies from 100 to 10,000 units: the equilibrium price just shifts accordingly. However, too little initial currency (10 units) led to liquidity problems, in that many exchanges executed at a 0 price, which led to indistinguishable performance between freeloaders and non-freeloaders; too much currency (1M units) took too long for price convergence, leading to the same result.

Experimental results. We now seek to demonstrate that our system fulfills the four hypotheses discussed at the beginning of the section.

Figure 6 plots the prices of sell clients' local (top) and remote (bottom) links. Given their smaller capacity, remote link prices remain higher for longer. Once chunks are disseminated to a few peers per cluster, local supply can largely satisfy local demand, and local prices drop. We can observe convergence around the remote equilibrium price between time 30 and 50 (the jagged edge corresponds to the MIMD oscillations around the equilibrium price).

Figure 7 gives these prices' performance implications. We observe four distinct regions: In the first ~ 18 time slots, the network is largely quiescent, as peers are still slowly arriving and beginning transmissions. Between 18–27, the rate of

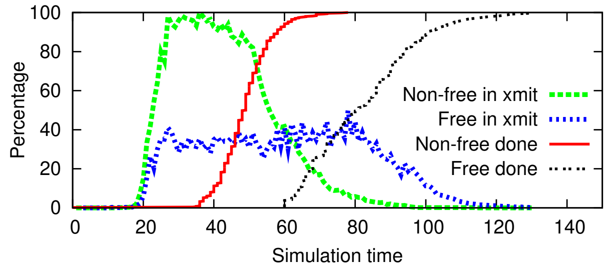

Figure 7: **Non-freeloaders (dashed green on left) download chunks earlier than freeloaders (dotted blue), leading to earlier completion times for non-freeloaders (solid red) than freeloaders (dotted black).**

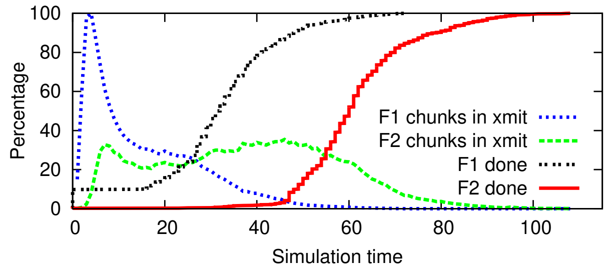

Figure 8: **Transfer rates for two files in the same network. File 1 starts at 50 nodes; file 2 at 1 node.**

chunk transmissions for both non-freeloaders and freeloaders greatly increases, as prices are still low (see Figure 6) and every node still has starting capital: The system has yet to reach an equilibrium. Already between time 21–60, however, we see the benefit of cooperation: Non-freeloaders download significantly more chunks than freeloaders, as when prices are non-zero, only non-freeloaders accrue capital to afford such. Finally, around time 50, non-freeloaders begin to finish, and prices drop toward zero given the enlarged supply. Freeloaders take advantage of this excess capacity (as is socially efficient) and their transmissions increase again. The result: non-freeloaders complete in about 60% of the time.

Figure 8 demonstrates the system dynamics for multiple files in more flash-crowd scenarios (*i.e.*, all peers arrive at time 0). Here, we initiate our 500-node network (no freeloaders) with two files, F1 and F2. Given increased supply of F1 exists—it starts on 10% of nodes—its transmission rate initially shoots up. While the nodes initiated with F1 start downloading F2 immediately (F2's initial bump in "chunks in transit"), most nodes delay while they download F1: F1's supply and lower available prices have led to saturated downstream links. As individual nodes finish downloading F1, they begin to download F2. Resource allocation thus properly adapts to constraints, yielding a median completion time for F1 that is 50% of F2. Under the Poisson arrival setup used earlier, the dissemination of both files behaved more identically, given that the lower initial demand on F1 allowed continued dissemination of F2.

Finally, Figure 9 shows that most transfers are local. We

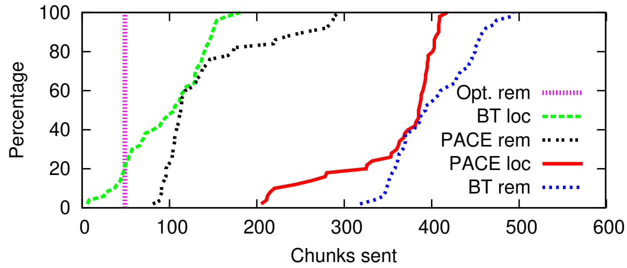

Figure 9: **PACE downloaders prefer local (*loc*) to remote (*rem*) transmissions, but BitTorrent (*BT*) makes poor use of locality.**

return to the single-file case, again with no freeloaders. The graph plots a CDF of chunk transmission frequency (*i.e.*, how many times each chunk traverses local and remote links). The minimum number that each chunk needs to be remotely transfered is 49, which would lead to 450 local transfers (given 50 clusters, 500 users, and 1 initial publisher). Our graph shows a median number 131% more remote transfers, primarily caused when chunks sent at low rates (and hence multiple rounds) are concurrently downloaded by multiple peers in the same cluster. Conversely, median local transfers occur at 86% of optimal.

Our BitTorrent simulation, however, performed significantly worse. Figure 9 shows BitTorrent's remote transmissions were 702% greater than optimal, or 26% of the optimal for local. One could add some static locality to BitTorrent [7, 1], but these do not capture dynamic constraints. Adapting to these dynamic constraints is useful; recall that Figure 1 shows PACE outperforming BitTorrent when it had to allocate resources across files. We omit further comparisons between PACE and BitTorrent due to space limitations.

9. RELATED WORK

Early P2P systems did not provide any incentives for participation, leading to extensive freeloading. According to [14], 85% of Gnutella users were sharing no files. The P2P community responded with mechanisms to prevent freeloading by incentivizing sharing.

One approach is to design a system based on *bilateral barter*, as used by BitTorrent [8] and its variants [24, 31], where users can achieve better download performance from peers to which they are simultaneously uploading. Not only are there no network considerations, but users are also not incentivized to continue uploading a file after they finish downloading it, making such systems ill-suited for anything but flash crowds for very large files. Our system encourages uploading, as this builds a user's budget for future downloads. Finally, [25] proposes a volume-based tit-for-tat that is coordinated through a relatively small set of intermediaries.

Another option is monetary incentives [13, 29]: A user's budget decreases when downloading a file and increases when uploading. MojoNation allowed users to price individual transactions in a centralized auction, but the usability hurdle for doing so—which is instead hidden from users in our design—was seen as its downfall [30]. Dandelion [27] describes currency-backed exchanges that use an online centralized bank, but gives no consideration about the resulting market, *i.e.*, how prices are set or adapt. Kash *et al.* studies performance as a function of the total amount of internal currency available [17], but does not consider heterogeneity in the system. While the market-theoretic formulation of [4] considers files with different prices, it does not propose a system design. Further, none of these approaches consider efficient resource utilization; in particular, no pricing is used for communication constraints between peers.

10. CONCLUSIONS

This paper studies the role of prices in peer-assisted content distribution. Our novel theoretical results demonstrate how relatively simple pricing mechanisms are sufficient for efficient allocations. Given these results, we present PACE, a system design for currency-backed content exchange. Beyond efficient use of network resources, PACE's algorithmic mechanisms are promising to hide complexity from users, provide robustness to strategic deviations, incorporate network friendliness, and prevent cheating.

Beyond a more in-depth consideration of ISP-propagated network prices, two other aspects for future work are interesting: long-term money management and the role prices could play in server provisioning. As users join and leave, and the network size changes, we must ensure that an appropriate amount of currency remains in the system. While the MIMD adaptation of prices makes the system somewhat robust to moderate inflation and deflation, excessive inflation may cause price convergence to take too long, while excessive deflation can lead to insufficient liquidity. We currently suggest a proportional "tax" on client's balances to prevent hoarding, and reinjecting money into the system via equal rebates to all parties. We leave a careful analysis of money management and rebating schemes to future work.

Content providers seek to use peer-assisted content distribution to cut costs, yet they also seek a certain quality-of-service. In PACE, providers can use prices, much like users do, to allocate their server resources near optimally. An interesting question remains how providers can use these prices to determine the level of provisioning sufficient to achieve desired QoS guarantees.

Acknowledgments. This paper was supported by the National Science Foundation.

11. REFERENCES

[1] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P users cooperate for improved performance? *ACM CCR*, 37(3), 2007.
[2] C. Aperjis, M. J. Freedman, and R. Johari. A comparison of bilateral and multilateral exchanges for peer-assisted content distribution. In *NetCoop*, Sept. 2008.
[3] C. Aperjis, M. J. Freedman, and R. Johari. The role of prices in peer-assisted content distribution. Technical Report TR-814-08, Princeton University, Computer Science, 2008.
[4] C. Aperjis and R. Johari. A peer-to-peer system as an exchange economy. In *GameNets*, 2006.
[5] E. Bangeman. P2P responsible for as much as 90 percent of all 'Net traffic. *ArsTechnica*, Sep 3 2007.
[6] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *EUROCRYPT*, 1996.
[7] R. Bindal and P. Cao. Can self-organizing P2P file distribution provide QoS guarantees? *OSR, Self-Organizing Systems*, 2006.
[8] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.
[9] J. R. Douceur. The Sybil attack. In *IPTPS*, 2002.
[10] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *NSDI*, 2004.
[11] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for any service. In *NSDI*, May 2006.
[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *IMC*, 2005.
[13] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV*, 2003.
[14] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on Gnutella revisited: The bell tolls? *IEEE Dist. Systems Online*, 6(6), 2005.
[15] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. In *WEIS*, 2005.
[16] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet service providers fear peer-assisted content distribution? In *IMC*, 2005.
[17] I. Kash, E. Friedman, and J. Halpern. Optimizing scrip systems: Efficiency, crashes, hoarders, and altruists. In *EC*, 2007.
[18] Q. Lian, Z. Zhang, M. Yang, B. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the Maze P2P file-sharing system. In *ICDCS*, 2007.
[19] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in BitTorrent is cheap. In *HotNets*, 2006.
[20] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.
[21] A. Mascolell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.
[22] A. Medina, A. Lakhina, I. Matta, and J. Byers. Boston University Representative Internet Topology Generator, 2007.
[23] T. Mennecke. The Pirate Bay breaks 10 million users. *Slyck News*, Jan 26 2008.
[24] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent? In *NSDI*, 2007.
[25] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer-to-peer file sharing workloads. In *NSDI*, 2008.
[26] M. Sirivianos, J. H. Park, R. Chen, and X. Yang. Free-riding in BitTorrent networks with the large view exploit. In *IPTPS*, 2007.
[27] M. Sirivianos, J. H. Park, X. Yang, and S. Jarecki. Dandelion: Cooperative content distribution with robust incentives. In *USENIX Technical*, 2007.
[28] Tribler. Bartercast. http://www.tribler.org/BarterCast, 2008.
[29] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A secure economic framework for P2P resource sharing. In *WEIS*, 2003.
[30] B. Wilcox-O'Hearn. Personal Communication, 2007.
[31] F. Wu and L. Zhang. Proportional response dynamics leads to market equilibrium. In *STOC*, 2007.
[32] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *SIGCOMM*, 2006.