Prices are Right:

Managing resources and incentives in peer-assisted content distribution

Michael J. Freedman*, Christina Aperjis†, and Ramesh Johari†
*Princeton University, †Stanford University

Abstract

We present a novel design for a peer-assisted content distribution system that addresses two key shortcomings of existing proposals. First, our system explicitly identifies the relative demand for files: users are rewarded for sharing more popular content. Second, our system efficiently utilizes network resources, by considering resource constraints explicitly when matching downloaders and uploaders.

Underlying our system is a market-based mechanism that enables the efficient allocation of network resources across multiple files. Although we price files and employ a virtual currency, these are purely an algorithmic ploy: the system clients hide the market details from the user. Nevertheless, our design also naturally incentivizes the contribution of scarce resources. More importantly, our design endogenously adapts peers' behaviors to changing environments, a critical advantage for real-world deployments in which network conditions, participation rates, resource demands, and content are continually in flux.

1 Introduction

Peer-to-peer (P2P) content distribution systems have been widely successful for scalable file sharing across the Internet, quickly delivering popular content to large numbers of users and reducing costs for content providers in the process. Traditional P2P designs have been overly focused on performance optimization, however, without addressing the overall welfare of participants. Scant attention is devoted to asking *which* files are "most useful" to disseminate, *where* resource congestion or network costs are accumulating, and *who* is providing useful content to the system.

This paper proposes a novel P2P system design that addresses these shortcomings. Its centerpiece is a market-based file-exchange mechanism that succinctly associates prices to files and resource constraints. This underlying market mechanism serves as a distributed algorithm that identifies and prioritizes popular content, while allocating scarce resources in a decentralized and efficient manner. Further, the mechanism securely incentivizes participation. We emphasize that our design employs a market-based solution primarily for algorithmic simplicity: our clients shield honest users from any notion of prices, budgeting, or other market issues, thus simplifying their experience.

At our system's core, users act as both *buyers* and *sellers*, with their software exchanging virtual currency in each transaction. As buyers, users specify which files they seek to download, while as sellers they decide which files they are willing to upload and their maximum aggregate upload rate. The system fully specifies the algorithmic buy-and-sell behavior of clients—*e.g.*, resource allocation and prices for sellers, and budgeting and peer selection for buyers. On the other hand, these underlying mechanisms prevent strategic or malicious clients from unduly damaging the system's efficient operation. In addition, a *network cost service* enables the inflation of wide-area traffic prices to accommodate network operator preferences, while a *rendezvous service* allows peers to discover one another and a *bank* tracks each users' accrued capital. We envision many cost, rendezvous, and bank services 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 a corpus spanning the entire web (as in CoralCDN [3]).

This paper makes three main contributions. First, we consider a new hierarchical network model that can be leveraged to satisfy the goals of all participants. This model captures the congestion points in a network and enables parties to price each accordingly. Our system employs decentralized pricing mechanisms; prior calls [1] for a single price per file not only require central agreement, but also ignore resource constraints *between* peers, which is critical for calculating supply and demand. By combining this network model with our pricing mechanisms, our system can flexibly and dynamically adjust to fit current operation conditions—across resources and files, whether typical or unexpected—and thus provides a form of environment robustness.

Second, users are *incentivized* to contribute to the system. Even though prices and budgets are invisible to users, each user's performance is affected by his budget. Thus, users must contribute upstream capacity to ensure a sufficient budget for downloading under resource constraints. On the other hand, if resources are not constrained, even non-contributors (freeloaders) can download files, as it is socially efficient for them to do. This design philosophy—viewing freeloading as a concern only in the context of resource constraints—diverges significantly from prior work.

Finally, our system architecture provides benefit to all major participants in a P2P system. *Users* benefit because our system efficiently allocates resources across the domain of many files, ensuring that only the most valued files are sent over expensive or congested links. *Network operators* can control impact on their networks, by setting "network cost multipliers" for long-haul traffic carriage. These costs ensure 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. *Content providers* benefit because prices can signal content providers as to how to allocate and provision scarce server resources (a significant area for future work). By recognizing these interests, we can turn P2P content distribution into a collaboration between all involved parties, as opposed to the contention that currently exists (*e.g.*, where ISPs curtail P2P traffic).

2 Previous Approaches

Early P2P systems did not provide any incentives for participation, leading to extensive freeloading. According to [6], 85% of Gnutella users were sharing no files. The P2P community responded with mechanisms to prevent freeloading, mainly focusing on incentivizing users to share content and upload capacity, but generally ignoring the *value* of content. Thus, Gnutella users who share unpopular files would still be freeloading, as their files would not be widely uploaded.

One approach is to design a system based on *bilateral peering* relationships. This is the case in BitTorrent [2], 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 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.

Another option is monetary incentives [5, 12]: A user's budget decreases every time he downloads a file, and increases every time he uploads. 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 [13]. Dandelion [11] 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 [7]. This approaches all use a single price for all files, thus ignoring heterogeneity in the value of files. While the market-theoretic formulation of [1] 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.

3 System Model and Design

3.1 System components

Our design creates a market for files, in which users act as both buyers and sellers. Software running on these users' end-hosts includes both a *buy client* and a *sell client*. These 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. This section describes the high-level design of these clients and other system components.

Buy and sell clients. Buy clients seek to achieve a good download rate across multiple files—which means allocating their budget for the users' desired files wisely—while sell clients seek to maximize their revenues (while maintaining a simple service discipline).

Our system accomplishes both tasks by pricing sell clients' resource constraints. A sell client s maintains a separate price for each file f—in "currency units per byte per second"—call this p_{sf} for now. As demand for f at s exceeds its available supply, s raises the price of p_{sf}; when supply exceeds demand, s lowers this price. We will extend this pricing model in §3.2, however, so that sell clients will price certain bottleneck links differently. §4 describes client behavior in more depth and why this design achieves efficient network usage and proper incentives.

Clients enforce the secure and fair exchange of content for virtual currency. In a transfer, the buy client initially commits to the sell client's price in a signed statement, which also includes the final output of a cryptographic hash chain. Then, as the sell client begins uploading content to the buy client, the buy client responds with a stream of micropayments, each a successive pre-image of the hash chain. This simple design ensures fairness: The buy client only sends micropayments for content actually received, while the sell client stops transmitting data soon after a buy client stops responding with micropayments. The buy client can thus "steal" bandwidth from at most a small transmission window before it is detected (and subsequently blacklisted locally by the sell client).

We securely manage clients' currency balances in a straight-forward manner: sell clients aggregate and deposit these currency transfers—*i.e.*, the signed commitment and final hash pre-image received—at a logically-centralized bank. This bank, described later, performs simple double-entry bookkeeping on the buy and sell clients' balances. The bank suitably penalizes or evicts any client maintaining a negative balance (akin to preventing currency forgery).

Network price service. Beyond minimizing resource congestion, our system benefits network operators by having clients avoid expensive network links whenever possible. Whenever a transfer would traverse the wide-area Internet, the sell client's file price is inflated by a *network cost multiplier* $\lambda \geq 1$, resulting in the total price λp_{sf}. As different λ are associated with different pairs of clients, the multiplier acts as a "shadow price" for resource constraints in the core.

When a payment from a buy to sell client is made for a transfer of duration t at rate r, however, only the amount $(p_{sf} \cdot t \cdot r)$ is paid to the seller. The remaining $((\lambda - 1)p_{sf} \cdot t \cdot r)$ is collected by the bank and then *rebated* back to all users (discussed a bit further in §6. This ensures that the sell client does not benefit from the additional payment due to network cost multipliers, which otherwise would create perverse incentives for the sell client to prefer uploading to expensive destinations. Note that ISPs or third parties do not receive any of this virtual currency.

To calculate the total price for downloading from a specific sell client, a buy client needs to discover the network cost multiplier between the clients. Various lookup service designs could provide buy clients with such information. Well-suited for immediate deployment, a third-party service (*e.g.*, OASIS [4]) could determine cost multipliers via network measurements. Alternatively, a buy client's network provider could run a lookup service itself (as with DNS and DHCP) and use these multipliers to perform traffic engineering on its egress traffic, taking its AS peering and transit relationships into account. In the longer term, we imagine such multipliers may be propagated *between* ASes, perhaps alongside BGP announcements, and thus enable ISPs to better express their preferences for P2P data transfers.

Although additive network costs initially appear natural, multiplicative network costs are more appropriate for this setting: With such, it is easier to give relative instead of absolute costs, and users can download without paying when file prices are zero, allowing the system to better leverage altruism (much like seeders in BitTorrent [2]). Multipliers should not be too high, however. Otherwise, a large fraction of a transfer's costs would go to the network; the subsequent rebate to users could provide freeloaders with sufficient currency to negate the incentive for uploading. Thus, a stand-alone service [4] should choose multipliers from a limited range, while multipliers propagated between ISPs should be normalized.

Rendezvous service. We have yet to prescribe how buy clients efficiently discover (nearby) sell clients offering their desired files. In practice, we envision that different instances of the system might employ various rendezvous techniques, from a logically-centralized tracker (a la BitTorrent [2]), to application-level anycast [4], to distributed hash tables indexing clients' addresses [3]. It is simply important that the rendezvous service provide both localized information—so that a buy client can discover nodes with low network cost multipliers—yet also sufficient diversity, in order to prevent individual sell clients from exerting market power (§4.3). Upon discovering a set of sell clients, the buy client contacts these nodes to determine both if they hold file chunks of interest and the file prices they offer.

Bank. Finally, a bank plays the role of maintaining users' account balances: a transfer of P_{bsf} from b to s requires subtracting money from b and awarding some part to s, with the (potentially zero) remainder from the network cost multiplier going to rebates. The light-weight means of payment verification—verifying a single signature and a chain of one-way hash functions—help minimize the bank's computational overhead.[1] This *logically*-centralized bank can be easily scaled by partitioning users over many servers (*e.g.*, via consistent hashing). Associated with the bank is a registration authority, which serves as a certificate authority for clients' public keys and seeks to make it difficult to operate many accounts per person (*i.e.*, the Sybil attack). Note that one benefits from multiple accounts only based on the systems' rebate policy; unlike with reputation or download/upload ratios [8], faking transfers between colluding nodes does not increase their aggregate balance.

We anticipate multiple network cost, rendezvous, and bank services to exist simultaneously. At its simplest, an entity that seeks to disseminate a *collection* of files creates a new currency, registers its users, and runs a bank for this collection. Thus, uploading any file in iTunes will result in "iTunes currency" to assist with any future download.

In fact, some settings might not require bank services at all. For example, locked-down set-top boxes for home entertainment systems might have trusted software implementing the buy and sell clients. Even though mechanisms for incentivizing clients may be unnecessary, the system's use of prices and virtual currency would still facilitate the efficient use of network resources.

3.2 Network model

Until now, we have described sell clients as maintaining a single price p_{sf} per file. This model does not leverage that

[1] We avoided anonymous e-cash, which require either online checks or offline exposable secrets. That said, our system is amenable to such.

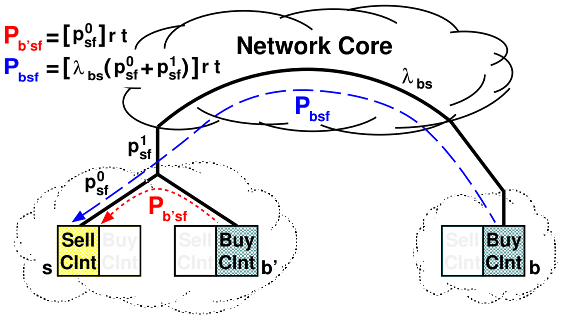

Figure 1: ***Network Model.*** **Clusters of well-connected peers connect across wide-area links. To download file f from sell client s, local buy client b' pays $P_{b'sf}$ and remote b pays P_{bsf} to s.**

sell clients may have varying network capacities (and hence supply) to different buy clients. Yet pricing every link in the network significantly increases complexity and leads to inaccurate bandwidth estimations. Instead, we propose a hierarchical network model, outlined in Figure 1, that captures most of the bottlenecks that lead to supply constraints.

A central observation of our network design is that, to a first approximation, we can view the entire network as composed of *local clusters*, connected together by the wide-area core network. 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 the slowly-changing network cost multipliers.

Our system design does not specify precise capacity requirements for local clusters and the wide-area network. Rather, 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).

Each file f at each sell client s has two prices, p^0_{sf} and p^1_{sf}, corresponding to the price on the seller's local and remote links, respectively. If buy client b' is within the same local cluster as s, b' pays $P_{b'sf} = (p^0_{sf} \cdot t \cdot r)$ to s. Such a payment is shown by the red dotted arrow in Figure 1. On the other hand, $(p^0_{sf} + p^1_{sf}) \cdot t \cdot r$ is paid to s if b is not in the same local cluster—the blue dashed arrow[2]—since any remote download must traverse both the immediate upstream link at s, as well as the access link shared by all nodes in the same cluster as s. The prices p^0_{sf} and p^1_{sf} are maintained and updated by s, in response to available upload bandwidth and the current demand for file f. In particular, if remote and local demand for f are both high, then the price p^0_{sf} will typically be less than p^1_{sf}: Even though remote traffic must also traverse the local link, local capacity is usually significantly higher than that of access links.

Buy clients always download a file's chunks from the sell clients providing the cheapest *total price* (see §4.2). When

[2] Any remote 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 b's downstream access link may not be correctly priced. This congestion applies to all remote files, however, and since local prices are typically lower, buy clients generally prefer local sources anyway.

a buy client requests a transfer from a sell client that is not in the same cluster, note that the traffic must also traverse the wide-area core, and is thus inflated by the network cost multiplier $\lambda \geq 1$. Thus, for any transfer of chunks of file f from a sell client s to a remote buy client b, the total price paid by b is $\left[\lambda_{bs}(p^0_{sf}+p^1_{sf})\right]$. Of this, only $(p^0_{sf}+p^1_{sf})$ is paid to the seller, with the remainder serving as rebates.

One could extend this same hierarchical model to capture additional choke-points at the network's edge and price these links separately and accordingly.

4 Mechanisms

4.1 Sell client

The sell client interface allows the user to declare (1) the files he is willing to upload (denoted by the set U_s for user s) and (2) the total upload capacity he is willing to commit (either as absolute numbers or as a percentage). For each file f, prices p^0_{sf} and p^1_{sf} are maintained by the sell client. This section details the service discipline and price update rules used by the sell client.

We start with the service discipline. Incoming download requests are not queued at the sell client. Requests are served in the order of arrival, subject to available capacity, and are immediately notified of acceptance. This greatly simplifies system design: If we queued requests, then buy clients would need to issue many requests in parallel, forcing them to optimize spending over a long time horizon.

With respect to price updates, the sell client follows a very simple rule in principle. However, a difficulty arises here as there are *two* scarce resources that the sell client is utilizing: his own immediate upstream link and the shared access link to the core—hence the two prices p^0_{sf} and p^1_{sf} for a single file. (A bottleneck at the access link of the buy client will be taken into account through demand, see §4.2).

Each sell client s estimates demand and supply for a file f over a fixed time interval. To update p^1_{sf}, demand is estimated as the total requested download rate originating at *remote* buy clients; a request is counted regardless of whether sufficient capacity existed to serve it.[3] To estimate supply, the possible upload rate is constrained by the sell client's estimate of available bandwidth on its access link; denote this estimate by c^1_s. Bandwidth estimation is used only for price adaptation, however: Users pay for actual content downloaded, even if throughput differs from the estimated rate. Thus, as we do not require *end-to-end* bandwidth estimation, available capacity may be approximated by tracking the maximum aggregate throughput ever seen across all transfers compared to current usage. (Note that systems such as BitTorrent similarly require edge capacity estimation, *e.g.*, to set its "active set size" [2].)

The sell client prefers to upload only those files with the highest current price p^1_{sf} (*i.e.*, the set of files $U^{\max}_s = \arg\max_{f\in U_s} p^1_{sf}$). Thus we set the supply of file f to zero if $f \notin U^{\max}_s$, and otherwise estimate available supply of f as $c^1_s/|U^{\max}_s|$. Prices increase if demand exceeds supply and fall otherwise; we use a multiplicative increase/multiplicative decrease rule, so that convergence is insensitive to the units in which prices are measured. When a sell client downloads the first chunk of a previously-unknown file, it sets its initial file price to that which it paid.

The approach to update p^0_{sf} 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 requests from local buy clients and the minimum of remote buy requests and the access link constraint. Supply is the available bandwidth on the sell client's immediate upstream link.

These forces simultaneously incorporate three effects: (1) the demand for files that the sell client has available, (2) the available capacity in the network, and (3) the sell client's preference to upload more expensive files. In this way, our system copes with both *heterogeneity* in the value of different files, as well as *efficient network utilization*.

4.2 Buy client

The buy client's interface allows the user to choose (1) the files he is interested in and (2) a *savings rate*, *i.e.*, the percentage η of the user's current budget that should be saved for the future. 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, order potential sell clients s in increasing order of the *total* price the buy client has to pay to s to download chunks of file f (including the network cost multiplier and remote access price, if applicable). Ties are broken in order of increasing network cost multiplier. Then, at each sell client s in this order, the buy client spends as much of its budget committed to f as possible. If s's upload capacity is exhausted, b moved to the next sell client in the list. The buy client stops when it exhausts its budget for f.

This algorithm, though quite simple, is based on a foundational utility model for the user. In particular, the buy client behaves *as if* a user's utility for downloading is:

$$\left[\ \sum_{\text{desired} f} \log(r_f) + \text{constant} \cdot \log(\text{saved budget})\ \right]$$

where r_f is the total download rate obtained for file f, and the constant depends on the saving rate η and the number of desired files. Maximizing this utility subject to the budget constraint faced by the buy client recovers exactly the algorithm described above.

4.3 Getting Incentives Right

This section argues that, in many respects, our system provides the correct incentives to participants. We begin by noting that, in particular, our system design encourages efficient use of resources in a *large* P2P system. If the system is large, we expect that it is hard for users to anticipate how their actions affect the prices, *i.e.*, it is difficult for users to predict how demand, supply, and prices will evolve in the future. One potential concern in such system is the phenomenon of *market power*: namely, that users may develop

[3] 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.

rogue sell clients 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 [9]. While market power may still be an issue if only a few users have a file, 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 disseminated, giving them no incentive to cheat. More generally, note that the seller *creates* other uploaders in the very act of uploading; thus, market power is at best a transient phenomenon, since other sellers quickly emerge as competitors.

Our system implicitly incentivizes sellers both to choose a high upload capacity and to share high-value content, since high-value files will typically be more profitable. This raises a potential problem: since serving remote clients is more profitable than serving local ones, rogue sellers may wish to prioritize uploads to remote clients. Yet there should be little tension between remote and local transfers in reality: assuming local capacity is much higher than access-link capacity, serving a remote client does not typically exclude a local one. We also note that the sell client is paid for delivered, not advertised, rate. Thus, a user does not profit by advertising a higher upload capacity than the one he has.

We also believe our sell client service discipline—serving requests sequentially and without preemption—is reasonably robust. A rogue seller might change this discipline, *e.g.*, rejecting requests even with available upload capacity, in the hope of getting requests for more expensive files in the near future. This is not significantly profitable, however, since chunks are relatively small. Moreover, if the system is large, predicting request arrivals is likely difficult.

Our system is designed so that the buy client chooses files consistent with its user's best interests, since it seeks to download at the current minimum price. Moreover, among files with the same local price, users prefer downloading locally; among remote transfers of the same price, they prefer the ones with the smallest network cost multiplier. Therefore, this aligns incentives with efficient network usage.

5 Simulation Analysis

This section evaluates 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) Buy clients prefer more efficient network links. (4) File prices yield efficient dissemination across multiple files.

Simulator design and configuration. We model the network using a hierarchical topology generated by BRITE [10], with clusters of nodes connected by AS-level links. In what follows, capacity between local hosts is 100 units/round, while that across wide-area links follows a heavy-tailed distribution $(10, 1024)$. Each file is comprised of 50 chunks, each of fixed size (25). These generated graphs yield pair-wise network capacities; for simplicity, however, we do not model cross-traffic congestion. We also assume simple fixed transmission rates (*e.g.*, no TCP slow state). Network cost multipliers are static and computed as an inverse logarithmic function of capacity, ranging $[1, 2)$.

The 5,000-line Python simulator operates in synchronous time steps. At each step, each buy client in random order allocates its budget over its desired files (per §4.2), choosing to download random chunks from each desired file from least-cost peers, subject to monetary and capacity constraints. Sell clients allocate their upstream supply and update their prices each round per §4.1.

We first evaluate the system's behavior for a flash-crowd for a single file: All users simultaneously become interested in the same file, initially at a single, random sell client (with price set to 1). The network is comprised of 500 peers, spread over 50 clusters of 10 nodes each. All nodes begin with 1,000 currency units. A randomly-chosen 50% of nodes are freeloaders, *i.e.*, they never upload content. Non-freeloaders upload—and thus accrue capital—after finishing their downloads. We conclude with the multi-file case.

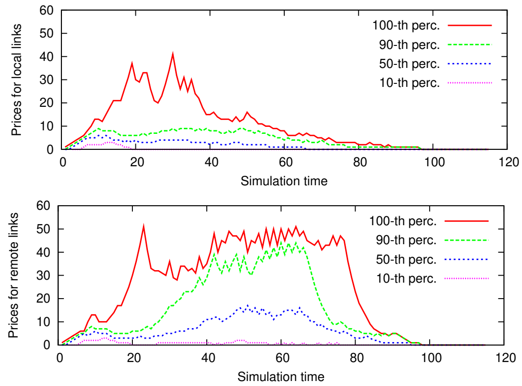

Figure 2: **Local links (top) are cheaper than remote (bottom).**

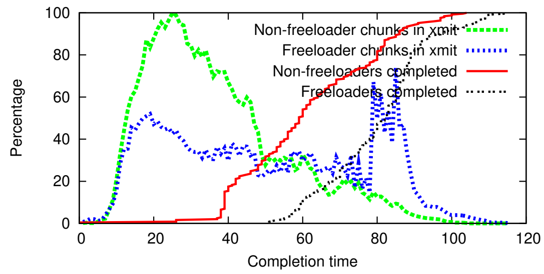

Figure 3: **Non-freeloaders (dashed green on left) download chunks earlier than freeloaders (dotted blue), leading to earlier completion times for non-freeloaders (solid red) than freeloaders (dotted black).**

Experimental results. We now seek to demonstrate that our system fulfills the above four hypotheses.

Figure 2 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 quickly drop.

Figure 3 gives these prices' performance implications. There are three distinct regions: In the first $\sim$20 time slots, the rate of transmissions for all parties greatly increases, as prices are still low (see Fig. 2) and every node still has starting capital. Between time 20–80, non-freeloaders download significantly more chunks than freeloaders: As prices remain non-zero, only non-freeloaders accrue capital to afford such. Finally, non-freeloaders largely finish; as prices drop

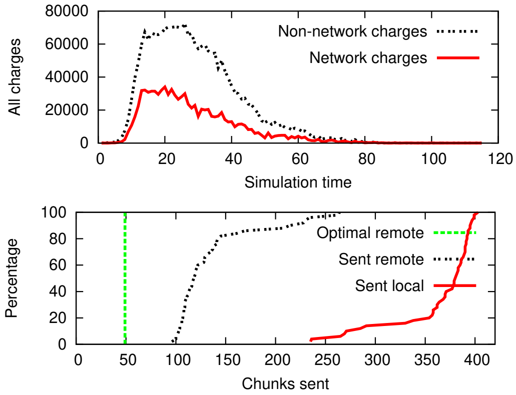

Figure 4: **Buyers prefer local transmissions.**

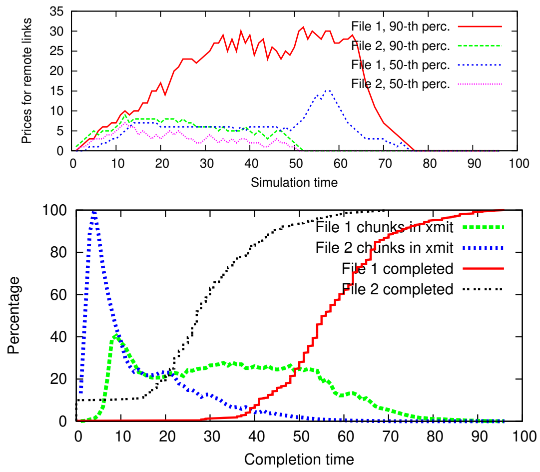

Figure 5: **Prices (top) and transfer rates (bottom) for two files in the same network. File 1 starts at 1 node; file 2 at 50 nodes.**

towards zero given the excess supply, the transmission rates of freeloaders quickly rise. The result: Non-freeloaders complete in about two-thirds the time.

Figure 4 shows that buy clients prefer to download locally. A much larger fraction of money goes to sell clients than to network costs (top graph). The bottom graph plots a CDF of chunk transmission frequency (*i.e.*, how many times each chunk traverses local and remote links). An optimal scheduler would transfer each chunk 49 times remotely and 450 times locally (given 50 clusters, 500 users and 1 initial publisher). Our graph shows a median number of remote transmissions 120% greater, primarily due to chunks sent at low rates (and hence multiple rounds) being concurrently downloaded by multiple peers in the same cluster. Thus, local transmissions occur at a median rate of 85% of optimal.

We also found that the completion time of all nodes appears to grow logarithmically with network size, ranging from an average of 59.0 time slots for 20 nodes to 156.2 slots for 1000 node systems, with only a small standard deviation between runs. All simulations used a cluster size of 10; we omit the graph due to space constraints.

Finally, Figure 5 demonstrates the system dynamics of multiple files. Here, we initiate our 500-node network (no freeloaders) with two files. Given that sufficient supply of file 2 exists, its price stays low and its transmission rapidly increases. While the nodes initiated with file 2 start downloading file 1 immediately (1's initial bump), most nodes delay while they download 2: File 2's low prices have led to saturated downstream links. After that time, 1 still lacks sufficient supply, and its price rises given the influx of demand from the nodes previously focusing on 2. Therefore, we see that prices adapt properly to resource constraints.

6 Conclusions and Future Work

This paper proposes a market-based system design that incentivizes users to share valuable content and use network resources efficiently. The design is novel in that it promises benefit for all participants: users, network operators, and content providers. We conclude by briefly discussing two directions for future work: the management of the money supply and server-side provisioning.

For our short time-scale evaluation, the total currency in the system was constant. In a real system, however, users may join (starting with some initial money or bootstrapped by "free" popular content, akin to BitTorrent's use of seeders), leave (dissipating money), and hoard money. If no monetary policy is enacted, the system may undergo inflationary or deflationary pressures. However, our logically-centralized bank is in a prime position to control the money supply, either through the rebate policy or modifying users' bank balancing (*e.g.*, by decreasing the value of all node's old currency, either proportionally or progressively, and giving an additional direct rebate at a regular rate).

Content providers seek to use peer-assisted content distribution to cut server costs, yet they often seek to provide some performance quality-of-service level to their users. In our model, as prices typically reflect the shortage of resources for a desired file, content providers can use prices, much like users do, to allocate existing resources near optimally. An interesting question remains, however, how to use these prices to determine the server or network provisioning sufficient to achieve a certain quality-of-service.

References

[1] C. Aperjis and R. Johari. A peer-to-peer system as an exchange economy. In *GameNets*, 2006.

[2] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[3] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *NSDI*, 2004.

[4] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for any service. In *NSDI*, 2006.

[5] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV*, 2003.

[6] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on Gnutella revisited: The bell tolls? *IEEE Dist. Systems Online*, 6(6), 2005.

[7] I. Kash, E. J. Friedman, and J. Y. Halpern. Optimizing scrip systems: Efficiency, crashes, hoarders, and altruists. In *EC*, 2007.

[8] 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.

[9] A. Mascolell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers. Boston University Representative Internet Topology Generator, 2007.

[11] M. Sirivianos, J. H. Park, X. Yang, and S. Jarecki. Dandelion: Cooperative content distribution with robust incentives. In *USENIX*, 2007.

[12] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A secure economic framework for P2P resource sharing. In *WEIS*, 2003.

[13] B. Wilcox-O'Hearn. Personal Communication, 2007.