On the Benefits of Cooperative Proxy Caching for Peer-to-Peer Traffic

Mohamed Hefeeda, *Senior Member*, *IEEE*, and Behrooz Noorizadeh

Abstract—This paper analyzes the potential of cooperative proxy caching for peer-to-peer (P2P) traffic as a means to ease the burden imposed by P2P traffic on Internet Service Providers (ISPs). In particular, we propose two models for cooperative caching of P2P traffic. The first model enables cooperation among caches that belong to different autonomous systems (ASs), while the second considers cooperation among caches deployed within the same AS. We analyze the potential gain of cooperative caching in these two models. To perform this analysis, we conduct an eight-month measurement study on a popular P2P system to collect traffic traces for multiple caches. Then, we perform extensive trace-based simulations to analyze different angles of cooperative caching schemes. Our results demonstrate that: 1) significant improvement in byte hit rate can be achieved using cooperative caching, 2) simple object replacement policies are sufficient to achieve that gain, and 3) the overhead imposed by cooperative caching is negligible. In addition, we develop an analytic model to assess the gain from cooperative caching in different settings. The model accounts for number of caches, salient P2P traffic features, and network characteristics. Our model confirms that substantial gains from cooperative caching are attainable under wide ranges of traffic and network characteristics.

Index Terms—Peer-to-peer systems, caching, cooperative caching, traffic modeling.

1 Introduction

Peer-to-peer (P2P) systems currently generate a major fraction of the total Internet traffic [1], [2], accounting for as much as 60-70 percent of the traffic in some Internet Service Providers (ISPs). Furthermore, it is expected that the amount of P2P traffic will even increase in the future [3]. Previous studies, e.g., [4], have shown that the huge volume of the P2P traffic multiplies the load on ISP networks and increases the possibilities of network congestion. Increasing the traffic load on ISP networks escalates the costs incurred by ISPs to provision and run their networks. These costs will, eventually, have to be passed to customers. In addition, since the Internet is a shared platform, higher chances of network congestion may indirectly degrade the performance of other Internet applications.

Several approaches have been proposed in the literature to reduce the impacts of P2P traffic. These include enhancing traffic locality [4], [5] and traffic caching [6]. More aggressive approaches using devices for traffic blocking and shaping have also been used in practice [7], [8]. These aggressive approaches, however, may not always be feasible for some ISPs because many of their clients like to participate in P2P systems and might switch to other ISPs if they were blocked. We believe that multiple approaches will likely be required to mitigate the problems created by the enormous amount of P2P traffic. For example, caching can be used in conjunction with locality-aware neighbor selection algorithms [5] to further reduce the amount of traffic downloaded from sources outside of the local network domain.

We focus on exploring the full potential of caching P2P file-sharing traffic (we refer to it simply as P2P traffic). In particular, we study the models, benefits, and costs of *cooperative* proxy caching for P2P traffic, where multiple proxy caches cooperate with each other to serve P2P traffic. Caching is a promising approach because objects in P2P systems are mostly immutable [1] and the traffic is highly repetitive [9]. In addition, caching does not require changing the P2P protocols and can be deployed transparently from clients. Therefore, ISPs can readily deploy caching systems to reduce their costs. In fact, several commercial P2P caching products have already made it to the market [10], [11], [12]. Efficient caching algorithms have also been proposed in the literature [6], [13]. However, all of these works and products are designed only for *independent* caches, i.e., caches that are neither aware nor cooperate with each other. Despite its great potential, as will be shown in this paper, cooperative caching for P2P traffic has received little attention in the literature. This is in contrast to the significant research attention that has been paid to cooperative caching of Web traffic, although its gain is only achieved under certain conditions [14], [15], and even in these cases, the gain may not be significant [16].

In this paper, we propose two models for cooperative caching of P2P traffic. The first model enables cooperation among caches that belong to different autonomous systems (ASs), while the second considers cooperation among caches deployed within the same AS. We analyze the potential gain of cooperative caching in these two models. To perform this analysis, we conduct an eight-month measurement study on the Gnutella P2P system to collect traffic traces for multiple caches. Then, we perform extensive trace-based simulations to analyze different angles of cooperative caching schemes. Our results demonstrate that: 1) significant improvement in

- *M. Hefeeda is with the School of Computing Science, Simon Fraser University, 250-13450 102nd Ave, Surrey, BC V3T 0A3, Canada. E-mail: mhefeeda@cs.sfu.ca.*
- *B. Noorizadeh is with Nokia, 9200 Glenlyon Parkway, Burnaby, Vancouver, BC V5J 5J8, Canada. E-mail: behroozn@gmail.com.*

Manuscript received 29 Feb. 2008; revised 18 Oct. 2008; accepted 30 July 2009; published online 5 Aug. 2009.
Recommended for acceptance by C. Shahabi.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-02-0080.
Digital Object Identifier no. 10.1109/TPDS.2009.130.

byte hit rate can be achieved using cooperative caching, 2) simple object replacement policies are sufficient to achieve that gain, and 3) the overhead imposed by cooperative caching is negligible. In addition, we develop an analytic model to assess the gain from cooperative caching in different settings. The analytic model complements our simulation study, and it addresses questions such as what happens if the P2P traffic characteristics and network conditions were to vary over wide ranges of possible values. The proposed model accounts for the number of caches, salient P2P traffic features, and network characteristics. By numerically analyzing the model, we confirm that substantial gains from cooperative caching are attainable under wide ranges of traffic and network characteristics.

The rest of the paper is organized as follows: In Section 2, we summarize the related work. In Section 3, we describe different models for caching P2P traffic. We present our measurement study in Section 4. In Section 5, we present trace-based simulation experiments to show the potential of cooperative caching. In Section 6, we propose and analyze several object replacement policies for cooperative caching. We also analyze the overhead introduced because of cooperation among caches. The analytic model is presented in Section 7. Finally, we conclude the paper in Section 8.

2 Related Work

Cooperative caching for Web traffic has been extensively studied, see, for example, [14], [15], [16] and the references therein. Using trace-based simulation and analytical analysis, Wolfman et al. [16] argue that cooperation yields marginal benefits for Web caching. Lee et al. [15] show that cooperation may be beneficial only in certain environments. The gain from cooperation in Web caching is debatable due to the following:

1. Web objects are fairly dynamic.
2. A Web proxy cache may be able to store most of the popular objects locally.
3. The overhead imposed is high relative to object sizes.
4. Latency could be increased due to looking up and downloading objects from other caches.

None of the above reasons exists in the case of cooperative caching of P2P traffic. First, most objects in P2P systems are immutable [1]. Second, because P2P objects are several order of magnitudes larger than Web objects [1], [13], it is unlikely that a single cache can hold a reasonable fraction of popular P2P objects to achieve high byte rate. The large object sizes in P2P systems also make the overhead paid to find and retrieve a requested object from other caches negligible. Finally, adding a few hundred milliseconds of latency to a P2P download is not a critical concern because many sessions take long periods (minutes and even hours) and they usually run in the background [1]. This is unlike Web sessions in which latency is crucial. Therefore, we believe that cooperative caching has a stronger case in P2P systems than it had in the Web.

Caching of the P2P traffic has recently been studied in a number of papers. The benefits of caching P2P traffic have been shown in [9] and [4]. Leibowitz et al. [9] show that P2P traffic is highly repetitive, and therefore, amenable to caching. The study in [4] suggests deploying caches or making P2P protocols locality aware to reduce the load on ISP networks. No caching algorithms were proposed in [9], [4]. Caching algorithms designed for P2P traffic have been proposed in [6] and in our previous work [13], [17]. In [6], two object replacement algorithms are suggested: Minimum Relative Size (MINRS) and Least Sent Byte (LSB). The first algorithm evicts the object which has the least cached fraction, and the second one evicts the object which served the least number of bytes from the cache. In our previous work [13], [17], we proposed a caching algorithm based on segmentation and partial caching of objects. We also designed and implemented a prototype for a proxy cache for P2P traffic [18]. Shen et al. [19] propose using already-deployed Web caches to serve P2P traffic. This, however, requires modifying the P2P protocols to wrap their messages in HTTP format and to discover the location of the nearest Web cache. Given the distributed and autonomous nature of the communities developing P2P client software, incorporating these modifications into actual clients may not be practical. Furthermore, previous works have shown that the object replacement algorithms designed for Web traffic may not yield good performance for the P2P traffic [13]. All the of above works, including ours, target independent caches and do not consider cooperation among caches to further enhance the byte hit rate.

Finally, several measurement studies have analyzed various aspects of P2P systems. Gummadi et al. [1] study the object characteristics of P2P traffic and show that P2P objects are mainly immutable, multimedia, large objects that are downloaded at most once. The study demonstrates that the popularity of P2P objects does not follow Zipf distribution, which is usually used to model the popularity of Web objects [20]. Sen and Wang [2] study the aggregate properties of P2P traffic in a large-scale ISP and confirm that P2P traffic does not obey Zipf distribution. Klemm et al. [21] use two Zipf-like distributions to model query popularity in Gnutella. While these measurement studies provide useful insights on P2P systems, they were not explicitly designed to study caching P2P traffic. They also did not provide traces that can be used with different caches. Therefore, we had to conduct our own measurement study.

3 Models for Caching P2P Traffic

3.1 Independent Proxy Caches

In independent caching, a cache is deployed near the gateway routers of ASs that choose to employ caching to reduce the burden of P2P traffic. See Fig. 1a, but note that caches in different ASs work independently from each other. In order to take full advantage of a deployed cache and to avoid modifying the source code of P2P client software, the cache should work in a *transparent* mode. This is similar to Web caching systems such as Squid [22], where the gateway router detects HTTP requests and forwards them to the Web cache. Detecting P2P traffic, however, is a bit more involved because many P2P systems use dynamic ports and some of them even encrypt control packets. Nonetheless, there have been several works on identifying

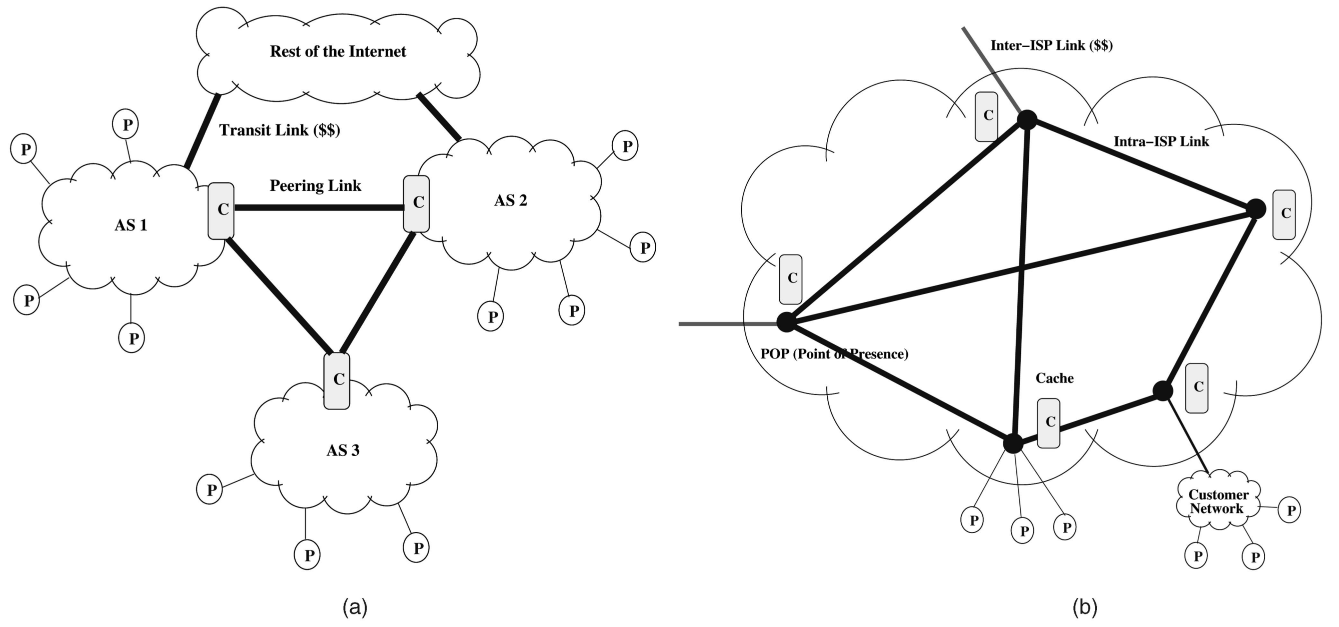


Fig. 1. The proposed two models for cooperative caching of P2P traffic. (a) Cooperation among caches in different ASs. (b) Cooperation among caches within a large ISP.

P2P traffic based on application signatures [23] and connection patterns [24]. Many commercially available P2P traffic shaping and blocking products, e.g., Packeteer [7] and P-Cube [8], already identify most of the packets belonging to P2P systems.

As argued in [6], [13], the primary goal of caching P2P traffic is to reduce the load on backbone links, and hence, reduce the operational costs of ISPs. To reflect this goal, we choose the *byte hit rate* as the main performance metric for evaluating caching systems for P2P traffic. The byte hit rate (BHR) is defined as the ratio of the number of bytes served from the cache to the total number of bytes transferred. Note that, unlike the case of caching Web traffic, the *hit rate*—defined as the ratio of the number of objects served from the cache to the total number of objects transferred—may not be well defined in the P2P case [6]. This is because requests in P2P systems are typically issued for *segments* of objects, not for entire objects.

3.2 Cooperative Proxy Caches in Different ASs

The first model for cooperation considered in this paper is depicted in Fig. 1a. In this model, caches deployed in different ASs cooperate with each other to serve requests from clients in their networks. The cooperating ASs may have a peering relationship to carry each other's traffic, or they can be located within the same geographical area such as a city where the bandwidth within the region is typically more abundant than the bandwidth on long-haul, intercity, links. As a concrete example, consider the Metropolitan Vancouver Area in British Columbia, Canada. Several universities serve this area, including SFU, UBC, BCIT, among others. Each university is a different AS with its own network. All university ASs are interconnected through a very high speed (gigabits per second optical links) network called BCNET. The bandwidth among university ASs is abundant. On the other hand, the university ASs are connected to the rest of the Internet through commercial ISPs such as Telus and Shaw. The links to the Internet have much smaller bandwidth (tens of megabits per second) and they cost significantly more money than BCNET. In this example, if the universities in Vancouver were to deploy caches for P2P traffic and enable cooperation among these caches over BCNET, they would achieve significant reduction in their bills for accessing the Internet.

Caches cooperating with each other form what we call a *cache group*. The cooperation in the cache group works as follows: When a cache receives a request for an object that it does not store locally, it first finds out whether another cache in the cache group has the requested object. If any of them does have the object, the object is directly served to the requesting client. If otherwise, the request is forwarded to external sources. Communication and object lookup inside the cache group can be done in several ways. For example, a centralized directory can be used, similar to the CRISP protocol [25], [26] proposed for cooperative Web caching. The lookup process is straightforward in this case and it requires only two messages. However, the directory is a single point of failure and it requires frequent updates from participating caches. We adopt distributed lookup methods. One distributed lookup method is using the Internet Cache Protocol (ICP) [27], [28], which is implemented on top of the open-source Squid Web cache [22]. We note that minor modifications to ICP will need to be made to support the P2P traffic case. For example, two fields should be added to the query messages of ICP to indicate the start and end of the requested byte range because clients in P2P systems request segments of objects, not full objects at once.

3.3 Cooperative Proxy Caches within the Same AS

The second model for cooperation proposed in this paper is for caches deployed within the same AS, as shown in Fig. 1b. This model is suitable for a large ISP with multiple access/exit points. The network of such ISPs is composed of multiple points of presence (POPs) interconnected with high-speed

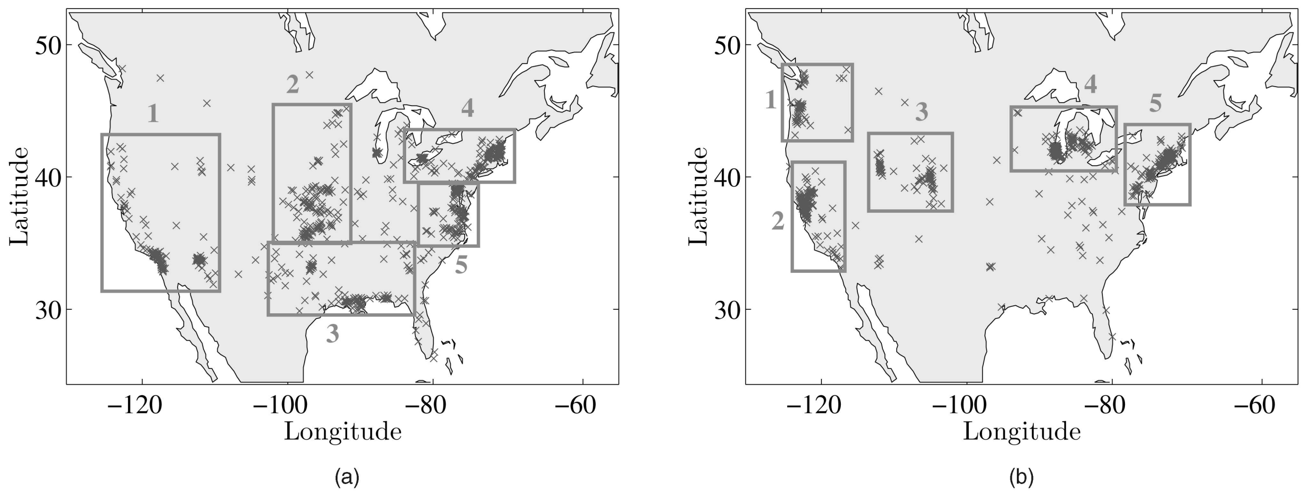


Fig. 2. Locations of peers in two large ASs in US. (a) Cox Communications (AS 11715). (b) AT&T-Comcast (AS 1859).

optical links. ISPs provide Internet access to their customers at POPs. The links inside an ISP are usually overprovisioned. ISPs are attached to the Internet through inter-ISP links. Inter-ISP links are usually the bottlenecks of the Internet and where congestion occurs. In addition, the inter-ISP links are expensive because an ISP either pays another ISP for carrying its traffic (in a customer-provider relationship) or it needs to mutually carry the same amount of traffic from the other ISP (in a peer-to-peer relationship) [29]. Deploying cooperative caches in such large ISPs would save a huge amount of P2P traffic from going on the inter-ISP links, and thus, would reduce the costs incurred by ISPs, because the cost of the internal links (between caches) is much smaller than the cost of inter-ISP links [30]. Caching would also benefit clients because their traffic will traverse fewer inter-ISP links, which are more susceptible to overload and congestion.

As concrete examples for this model of cooperative caching, we show in Fig. 2 the distribution of clients in two large ASs in US: AS 11715 (Cox Communications) and AS 1859 (AT&T-Comcast). We discuss how we created this map in Section 5. Since ISPs provide Internet access to their customers at POPs, they are the natural locations for deploying caches. Therefore, caches would be near client clusters, somewhere inside the rectangles in Fig. 2. Caches would cooperate to serve requests from P2P clients in the same AS in order to save traffic on expensive inter-ISP links. The cooperation among these caches employs protocols similar to the ones described in the previous section.

We note that cooperation among caches within the same AS would be easier to implement in practice than cooperation among caches in different ASs. This is because in the former case, all caches are owned and managed by a single entity, while in the latter case, multiple parties are involved. Moreover, political issues between different parties might affect the decision of enabling cooperative caching. Nonetheless, we hope that the significant potential gains shown in this paper will motivate ASs to enable cooperative caching. Finally, we mention that caching of P2P traffic might raise some legal issues, similar to those raised and addressed for caching of Web traffic about two decades ago. Discussing these legal issues is outside the scope of this paper.

4 Measurement Study and Trace Collection

We are interested in studying the potential collaboration among caches to reduce the WAN traffic imposed by P2P systems. Ideally, we would like to have a trace showing information about requested objects from each cache. Although several P2P caching products have already been introduced and deployed [10], [11], [12], we are not aware of any public traces that can be used to study caching of P2P traffic. Therefore, we conducted a measurement study on the Gnutella file-sharing network [31] to collect and analyze the characteristics of P2P traffic that would be observed by many individual caches.

Gnutella has two kinds of peers: ultrapeers, characterized by high bandwidth and long connection periods, and leaf peers, which are ordinary peers that only connect to ultrapeers. Peers exchange several types of messages including PING, PONG, QUERY, and QUERYHIT. We modified a popular Gnutella client, called Limewire [32], to run as a monitoring node. We ran our monitoring node in the ultrapeer mode. It passively recorded the contents of all QUERY and QUERYHIT messages passing through it without injecting any traffic into the network.

Although we deployed only one ultrapeer, we configured it to reach most of the Gnutella network as follows: We increased the number of concurrent connections that it can maintain to be up to 500. A regular ultrapeer allows up to 16 connections to other ultrapeers and up to 30 to leaf peers. Effectively, our peer was worth more than 20-30 regular ultrapeers. In many times, our peer was connected to more than 350 other ultrapeers. Let us assume that each of these 350 ultrapeers connects to other 10 ultrapeers, on average, each of them connects to other 10, and so on. Given that queries in Gnutella are forwarded up to 7 hops among ultrapeers, our peer was able to capture traffic from a huge number of peers. In addition, our peer ran continuously for eight months, while other peers joined and left the network. This means that the 200-300 other peers connected to our peer were continuously changing, which allowed our peer to reach different and larger portions of the Gnutella network.

The measurement study was conducted between 16 January 2006 and 16 September 2006. Our measurement

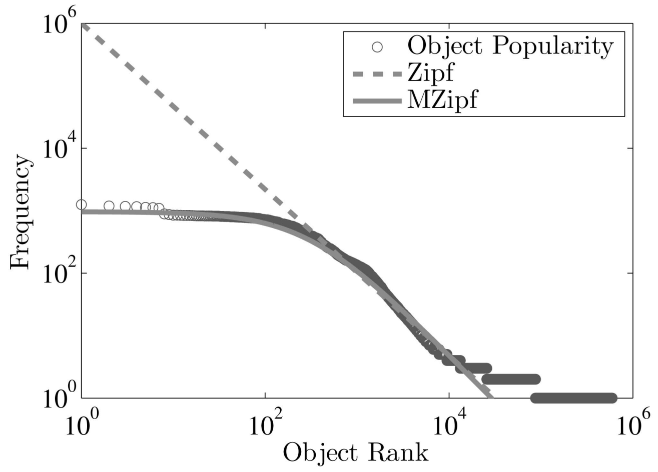


Fig. 3. Popularity of objects in P2P systems is better modeled by a Mandelbrot-Zipf distribution.

peer was located at Simon Fraser University, Canada. But since the Gnutella protocol does not favor nodes based on their geographic locations [21], we were able to observe peers from thousands of ASs across the globe. During the eight months of the measurement, we recorded more than 288 million QUERY messages and 134 million QUERYHIT messages issued from approximately 38 million peers distributed over more than 17 thousand different ASs. The large scale of our measurement study enables us to draw solid conclusions on the potential cooperation among caches for P2P traffic. We mention that these traces are rigorously analyzed in our previous work [13], which studies independent P2P caches with no cooperation among them.

We construct the traces for individual ASs as follows: When a peer replies with a QUERYHIT message for an object, it means that this peer has a copy of this object. This also means that most likely, this peer downloaded the object sometime in the past. Since most downloads are served by peers from outside the AS in which the client peer resides [1], a cache would have observed this download if it had been deployed at the gateway router of the client peer's AS. Thus, the trace for an AS is constructed by having one request for each replica downloaded by a peer in that AS. We use only the first QUERYHIT message sent by a peer for a particular object, which is identified by the source IP address and the object ID. Peers that replied earlier with QUERYHITs for an object are assumed to have downloaded the object earlier. Note that, from the cache perspective, the exact time when the object was downloaded is not important. It is the relative popularity of objects and the distance between similar requests in the trace that matter. These two issues are captured by our sequences.

An important aspect in caching P2P traffic is object popularity. The authors of [1] show that object popularity has a *flattened head*, and our previous work [13] confirms this flattened head nature and presents an analytic model for it. We illustrate this model in Fig. 3, which shows the object popularity in a sample AS from our traces. The figure shows that a Zipf-like distribution will overestimate the popularity of objects near the head of the curve, which is the most important for the cache. The figure also shows that a generalized form of Zipf-like distributions, called Mandelbrot-Zipf [33], is a better model for object popularity in P2P systems. The Mandelbrot-Zipf distribution defines the probability of accessing an object at rank i out of N available objects as:

$$f(i) = \frac{K}{(i+q)^{\alpha}}, \tag{1}$$

where $K = 1/(\sum_{i=1}^{N} 1/(i+q)^{\alpha})$, α is the skewness factor, and $q \geq 0$ is a parameter that we call the *plateau* factor, because it is the reason behind the plateau shape near to the leftmost part of the distribution. We will use the Mandelbrot-Zipf popularity distribution in our analysis throughout the paper.

TABLE 1
Summary Statistics for the P2P Traffic Observed by Our Monitoring Node in 10 ASs in the West Coast of North America

AS#	AS Name	Unique objects (TB)
2161	AT&T	30.41
9548	Road Runner	14.37
9406	Verizon	12.05
11394	Charter	14.99
11715	Cox	12.43
1859	AT&T-Comcast	59.67
1782	Shaw	27.96
233	Telus	19.02
18952	Comcast	17.51
105	Qwest	14.02

5 The Potential of Cooperative Caching for P2P Traffic

In this section, we use our traces to study various aspects of cooperative caching. We start by showing that cooperative caching is needed to achieve high byte hit rates and to save bandwidth on expensive links. Then, we demonstrate the potential gain from cooperation in the two models proposed in this paper.

5.1 The Need for Cooperation

We start our analysis by making the case for cooperative caching of P2P traffic. We choose 10 different ASs from our traces to see what would happen if each deployed a cache to serve P2P traffic originated from a given geographical region. As an example region, we select the West Coast of North America. In this region, we choose the 10 ASs with the largest amount of traffic seen in our traces to make our results statistically significant. For each of these 10 ASs, we find all requests issued from that AS. We use the IP addresses of requests to map a request to an AS using the GeoIP database, which is fairly accurate for cities in North America [34]. We refer to this process as IP-to-AS mapping. Since an AS can span multiple geographical regions, we need to remove requests from outside the West Coast. We do this by finding the geographical locations of the IP addresses of requests again using the GeoIP database (this is referred to as IP-to-geolocation mapping). Then, we remove all requests that are not issued from clients in the West Coast. Table 1 lists the names and summary statistics for these 10 ASs.

TABLE 2
P2P Traffic Statistics for Five ASs in Los Angeles

AS#	AS Name	Unique objects (TB)
2161	AT&T	18.86
9548	Road Runner	18.29
9406	Verizon	10.91
11394	Charter	7.72
11715	Cox	5.39

As Table 1 shows, the total size of *unique* objects observed in each AS is too large to fit in a single cache. Note that these statistics are for the data our monitoring node was able to capture from only one P2P system (Gnutella). The actual amount of P2P traffic for each AS is indeed much larger than that is shown in Table 1. Thus, the unique objects may not fit into one cache. In addition, as indicated by the flattened head of the object popularity distribution in Fig. 3, the probability of accessing objects is not concentrated in a few objects as in the Zipf-like distribution case. Rather, it is spread across a much larger number of objects. This means that a single cache may not be able to store enough popular objects to achieve a high byte hit rate. We contrast the need for cooperative caching in the P2P traffic case against that need in the Web traffic case. Previous studies, e.g., [16], indicate that an individual cache could store most of the cacheable Web objects. This is because Web objects have relatively small sizes, and many Web caching products limit the size of objects that can be stored locally.

In summary, given that individual caches may not have enough capacity to store popular objects to achieve high byte hit rate and that the P2P traffic can easily tolerate the small delay that might result from cooperative caching, we believe that cooperation among caches is *needed* to enhance the byte hit rate and save more traffic from going on expensive WAN links. It remains to see how much gain we could achieve from cooperative caching, which we study in the following two sections, and what costs are involved, which we analyze in Section 6.3.

5.2 Gain from Cooperation among Caches in Different ASs

We consider cooperation among ASs within the same city. We select two large cities: New York City (NYC) and Los Angeles (LA). In each city, we choose five popular ASs for the analysis. Each AS is assumed to deploy a cache, and the five caches form a cache group. Caches in the same group cooperate with each other to serve P2P traffic originated from the city in which they are located. Traces for caches in different ASs are created using the same IP-to-AS and IP-to-geolocation mapping methods explained in Section 5.1. Summary statistics about the five ASs in LA are listed in Table 2.

To assess the potential of cooperative caching, we conduct several simulation experiments to compare the byte hit rate in the cooperative caching case versus the byte hit rate in the independent caching case. We analyze the performance of the two cache groups in LA and NYC. We consider two realistic (online) replacement policies: cooperative Least Recently Used (denoted by cLRU) and cooperative Least Frequently Used (denoted by cLFU). Their counterparts in the independent caching case are referred to as iLRU and iLFU, respectively. cLRU (and similarly cLFU) works as follows: When a cache observes a miss and the object is found in another cache in the group, the object is downloaded from that cache but it is not stored locally. In this case, we have only one copy of any object within the cache group. The remote cache updates its data structure as if it had a hit from a local client. In this manner, all caches in the group cooperate to implement a group-wide LRU policy.

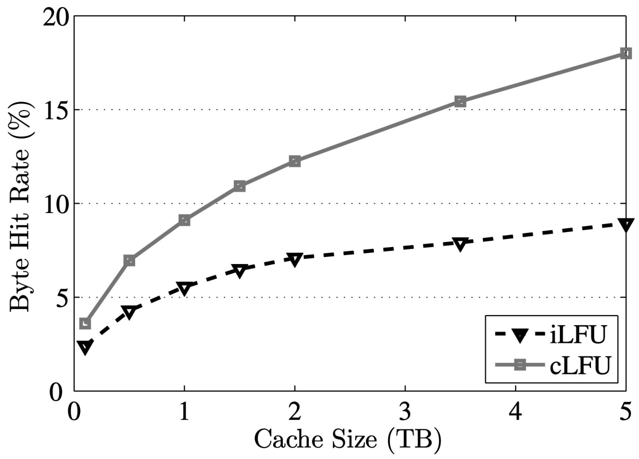


Fig. 4. Comparison between independent and cooperative caching for caches deployed in five ASs in Los Angeles. Sample results are shown for AS 2161.

We vary the size of individual caches from 100 GB to 5 TB, and we compute the byte hit rate achieved by iLRU versus cLRU and iLFU versus cLFU for each cache in the two cache groups. A sample of the results is shown in Fig. 4, other figures are similar. The figure implies that significant gains in byte hit rate can be achieved by cooperation among caches. For example, with a cache of size 2 TB, the byte hit rate achieved by cLFU is almost double that achieved by iLFU. In addition, the gain improves as the cache size increases, which is expected in the future as storage prices keep dropping. Thus, cooperation among caches is beneficial even when ASs deploy caches with large storage systems (several terabytes) because of the huge of amount of the P2P traffic.

In another experiment, we fix the cache size at 1 TB and compute the potential improvement in byte hit rate due to cooperation among caches in the same group. We compute the percentage of improvement in byte hit rate, which is the difference between the byte hit rates of cLRU and iLRU normalized by the byte hit rate of iLRU. We repeat the experiment for cLFU and iLFU. The results for the cache group in LA are given in Fig. 5; the results for the cache group in NYC are similar. The figure shows that up to 330 percent improvement in byte hit rate can be achieved for some ASs. The average gain across all five ASs is more than 120 percent. Fig. 5 also indicates that different replacement policies may yield different gains, and more importantly, some ASs may benefit more than others from cooperation. We elaborate more on this important issue in Section 6.2.

The experiments in this section show that the byte hit rate could be doubled or tripled in some cases because of cooperation. Considering the huge volume of the P2P

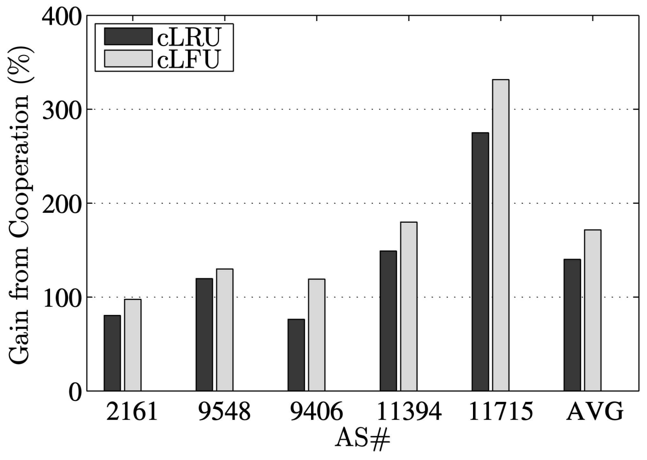


Fig. 5. Gain from cooperation using cLFU and cLRU replacement policies for five ASs in Los Angeles.

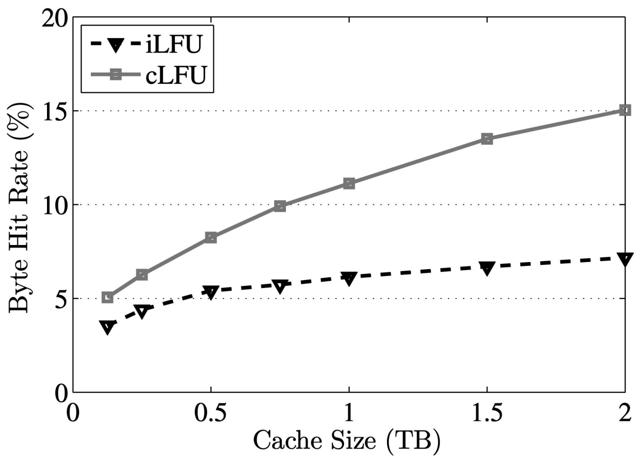


Fig. 6. Comparison between independent and cooperative caching for caches deployed in the same AS. Sample results are shown for AS 11715.

traffic, even 1 percent improvement in byte hit rate accounts to saving in the order of terabytes of traffic on the expensive WAN links. Therefore, the large savings from cooperation will serve as an incentive for ISPs to deploy caches and enable cooperation among them.

5.3 Gain from Cooperation among Caches within the Same AS

In this section, we study the potential gain from cooperation between caches deployed within the same AS. We choose several large ASs with clients distributed over many locations in North America. For each AS, we use the GeoIP database to map the IP addresses of clients in that AS to their geographical locations. The GeoIP database returns the latitude and longitude of a given IP address. We use Matlab to plot these values on the map of North America. Fig. 2 shows the distribution of clients in two large ISPs in USA. As mentioned in Section 3.3, the POPs of an AS are the usual locations for deploying caches. The exact locations of ISP's POPs, however, are not public information.[1] Therefore, we had to approximate the locations of major POPs. Intuitively, a POP will be near the clustering of many clients. In Fig. 2, we draw rectangles around apparent clustering of clients in our traces. We assume that the AS deploys a cache somewhere in each of these rectangles. Then, we create traffic traces for each cache by considering only requests from clients falling inside the rectangle in which the cache exists. Note that the approximate locations of caches do not affect the analysis of cooperative caching because they could only change the delay between a cache and its clients by a few milliseconds, which is a negligible effect in P2P traffic that runs in the background for much longer periods (minutes).

Similar to the previous case, we study the gain from cooperation under cLFU and cLRU policies versus iLFU and iLRU policies, respectively. Some of the results are shown for AS 11715 in Fig. 6. The results confirm that the byte hit rate could be significantly increased with cooperation among caches. Therefore, we can conclude that cooperation improves byte hit rates in both cooperation models considered in this paper.

1. We are aware of the Rocketfuel project [35], [36], which infers ISP topologies. We did not, however, find the topologies of the ISPs considered in our analysis in the data available from [36].

6 Replacement Policies and Cooperation Overhead

A replacement policy is used when a cache needs to evict an object (or a few objects) to make room for a newly requested one. The replacement policy is an important component of any caching system, and it is specially so for a P2P cache because of the large size of objects, and therefore, the limited number of them that a cache can store. In addition, replacement policies not only affect the total byte hit rate of the caching system, but they also impact the relative gain of individual caches in the cache group, as we will demonstrate in this section. Furthermore, different replacement policies impose different amounts of overheads, which are important to analyze in order to assess the net benefits of cooperative caching.

In this section, we first describe and analyze various replacement policies for cooperative caching. Then, we study the effect of the replacement policies on the gain achieved by different ASs and evaluate the overhead imposed due to cooperation among caches.

6.1 Replacement Policies for Cooperative Caching

We have already described two online replacement policies for cooperative caching in Section 5.2: cLFU and cLRU. cLFU and cLRU implement group-wide LFU and LRU policies, respectively. The authors of [6] have proposed a few replacement policies designed for caching P2P traffic. The LSB was shown to outperform other policies proposed in [6]. LSB works for individual caches and it evicts the object that has the minimum number of bytes transmitted from the cache. We consider the cooperative version of LSB, which is denoted by cLSB, as a candidate policy for cooperative caching. cLSB implements a group-wide LSB. All of the cLFU, cLRU, and cLSB replacement policies try to increase the total byte hit rate across all caches. In that sense they are global in nature. Therefore, they may evict locally popular objects from their caches if the global popularity of these objects is not high compared to other objects in the cache group. That is, the byte hit rate of some caches might be sacrificed for enhancing the total byte hit rate of the whole cache group. This uncertainty in the gain from cooperation might discourage ASs from enabling cooperation among caches.

To mitigate this problem, we propose a simple model for object replacement in cooperative caching. We call this model cooperative caching with *selfish* replacement. Under this model, a cache cooperates by serving requests issued from other caches in the cache group if it has them. The object replacement policy, however, bases its decision to evict objects only on local information of individual caches. We apply the selfish model on the three object replacement policies described above. This results in three new policies: sLFU, sLRU, and sLSB, where the prefix "s" means that the policy is "selfish." For example, a cache running sLRU replaces objects that have not been requested for the longest period of time from its clients, i.e., clients from the AS in which the cache is deployed.

We use our traces to analyze the performance of different replacement policies. We implemented the six policies described above: sLRU, sLFU, sLSB, cLRU, cLFU, and cLSB in our cooperative caching simulator. For each policy, we run four simulation experiments: 1) two for cooperation among caches in different ASs (the five caches in LA and the five caches in NYC) and 2) two for cooperation among caches within the same AS (AS 11715 and AS 1859). Therefore, we have a total of 24 simulation experiments, and each is run on an eight-month trace of requests. We study the replacement policies along multiple performance metrics. First, we consider the total byte hit rate achieved by each cache, which is defined as the number of bytes served from any cache in the group (including the local one) over the total number of bytes requested by the clients behind that cache. Then, we differentiate between bytes served from the local cache and bytes served from other caches in the cache group. We make this distinction because bytes served from other caches typically cost more (in terms of bandwidth and latency) than bytes served locally. We use L-BHR to refer to the byte hit rate achieved by serving objects only from the local cache, and G-BHR to refer to the byte hit rate from the whole cache group excluding the local one. Clearly, the total byte hit rate is the summation of L-BHR and G-BHR.

A sample of our results for the cache group in LA is shown in the bar charts in Fig. 7. The figure shows the average byte hit rates (L-BHR and G-BHR) for the five ASs in LA. Similar results were obtained for the other cache groups in NYC, AS 11715, and AS 1859. The results shown in Fig. 7 indicate that cLRU outperforms all other policies in terms of the total byte hit rate (L-BHR + G-BHR). However, the simpler (selfish) sLRU is not too far from it. In fact, sLRU is better in terms of local byte hit rates, which are more valuable. The reason that both LRU versions perform well is that the P2P traffic observes a good degree of temporal locality [17], as popular objects tend to stay popular for some time, then they gradually lose popularity.

6.2 Replacement Policies and Relative Gain from Cooperation

We noticed in Section 5.2 that some ASs may benefit more than others from cooperation. The experiments in that section are limited to only five ASs and they are all fairly large in terms of the amount of traffic seen in each AS. In

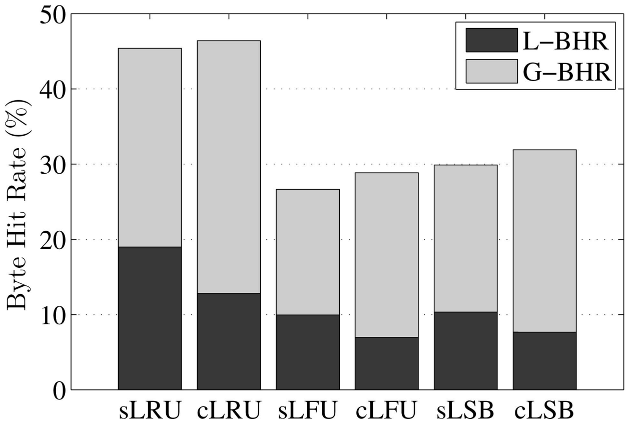


Fig. 7. Comparison among six replacement policies in terms of achieved group (G-BHR) and local (L-BHR) byte hit rates when cooperation is done among caches in different ASs in Los Angeles.

this section, we expand the level of cooperation to include 64 ASs with different sizes and we study the relative gain of each AS. For each of these 64 ASs, we create a trace file that contains requests observed in that AS. Then, we determine the number of objects seen in each trace file. We rank the ASs based on the number of objects that are requested in their corresponding trace files.

We simulate a cooperative cache group that contains all of the 64 ASs. We repeat the experiment several times, and each time, we use a different replacement policy to compute the gain in byte hit rate for each AS. For example, if cLFU is used, we measure the byte hit rate achieved by each AS when it cooperates with the others. We also measure the byte hit rates when ASs do not cooperate with each other, but each of them deploys a cache that uses iLFU. We compute the gain in byte hit rate due to cooperation (cLFU-iLFU) and normalize it by the byte hit rate of iLFU. We summarize the results in the scatter diagram in Fig. 8, where the x-axis represents the number of objects seen in an AS and the y-axis represents the percentage of improvement in byte hit rate observed by that AS because of cooperation.

Our results, two samples of them are shown in Fig. 8, imply that: 1) the selfish replacement policies (sLRU, sLFU, and sLSB) sort of equalize the relative gain from cooperation across all ASs and 2) among the selfish policies, sLRU produces the smallest improvement gap between ASs. For example, comparing Fig. 8a versus Fig. 8b indicates that there are a fewer number of ASs that achieve gain less than 10 percent under sLRU than under sLFU. Moreover, most of the dots representing gains of different ASs for sLFU in Fig. 8b are spread over a larger range of the y-axis than they are for sLRU in Fig. 8a. We note that the few ASs that did not gain much from cooperation are the ones with few objects (concentrated at the lower left part of the figure). These ASs achieved relatively high byte hit rates with independent caching, and thus, the relative improvement from cooperation was not significant. Therefore, cooperation may not be beneficial for ASs with small amount of P2P traffic.

In summary, the experiments in this section show that sLRU achieves high byte hit rates and produces the smallest differences in byte hit rate improvement among ASs with different sizes. In addition, as will be shown in the next

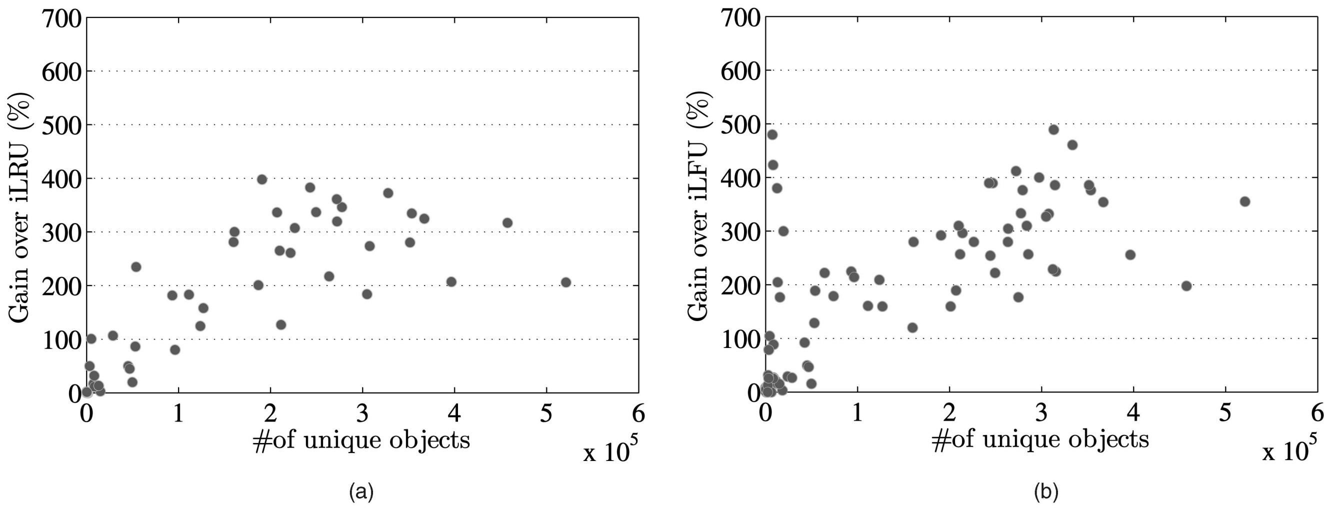


Fig. 8. Relative gain in byte hit rates when 64 ASs of different sizes cooperate with each other. (a) sLRU Gain. (b) sLFU Gain.

section, sLRU imposes the least amount of overhead. Therefore, we believe that the simple sLRU replacement policy is a good candidate for realizing the potential benefits of cooperative caching for P2P traffic.

6.3 Overhead Analysis in Cooperative Caching

Analyzing the overhead imposed by cooperative caching schemes is critical in understanding the *net* benefit of employing these schemes [14]. In cooperative Web caching, the overhead is one of the factors that plagued its wide deployment. We show below that this is not the case in cooperative caching for P2P traffic.

By overhead we mean the additional number of bytes transmitted beyond the transfer of the requested objects themselves. As mentioned in Section 3.2, we use the ICP [27], [28] to facilitate communication and object lookup among caches. We have implemented the ICP protocol in our cooperative caching simulator. We compute the overhead imposed by different replacement policies. As before, we consider the two caching groups in LA and NYC and the two caching groups within AS 11715 and AS 1859. We count the number of bytes that are exchanged by the ICP protocol and divide that number by the total number of transferred bytes. The results for the cache group in LA are shown in Fig. 9. The figure implies that the maximum overhead imposed by cooperative caching is less than 0.003 percent for all policies, which is indeed negligible. The figure also shows that sLRU has the smallest overhead. This is because sLRU has higher local byte hit rate (L-BHR), as discussed in the previous section. Local hits do not impose overhead because they do not require sending ICP queries to other caches.

7 Analytic Model

In this section, we analyze the benefits of cooperative caching for P2P traffic using a simple analytic model.

7.1 Model Parameters and Assumptions

The model captures the most important parameters in the caching system, including object popularity, number of caches, and the relative cost of network links inside an AS and across ASs. The model considers m caches deployed in different ASs and cooperating with each other. We refer to these m caches as a cache group. We use abstract costs in our analysis to make the model more general. For example, if the costs are set as delays, the model can be used to analyze the average latency perceived by clients (as it is usually done in Web caching). On the other hand, if we set the costs as dollars per bytes, the model would allow us to analyze the average saving in the operational costs observed by an AS because of cooperative caching. Analyzing the saving is more relevant to ASs that are interested in deploying caches to reduce the burden imposed by the sheer volume of the P2P traffic.

We denote the cost of retrieving an object by a client from: 1) its local cache by τ_l; 2) another cache in the group byτ_g; and 3) an external source (i.e., other peer(s) in the P2P network) by τ_e. The total number of objects in the system is denoted by N. The relative popularity of an object i is given by $f(i)$, which follows the Mandelbrot-Zipf model described in Section 4.

For the feasibility of the analysis, we make the following assumptions: In Section 7.4, we relax most of the assumptions used in deriving the analytic model and validate our analysis

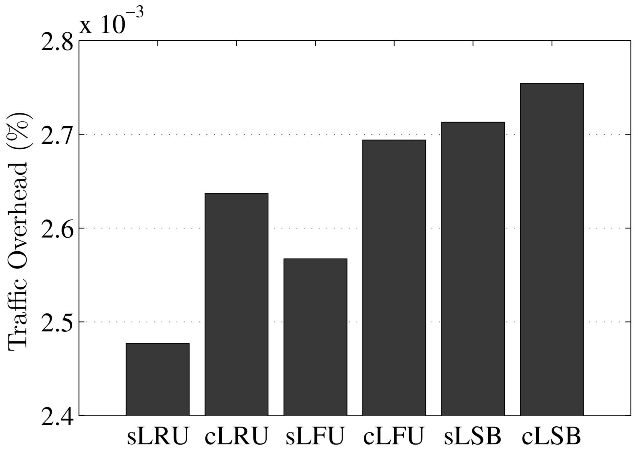


Fig. 9. Comparison among six replacement policies in terms of traffic overhead. Data are shown for five cooperating ASs in Los Angeles.

using simulation. First, all caches have the same storage size. Second, since objects in P2P systems are divided into equal-sized segments, we carry out our analysis in terms of segments. BitTorrent, for example, typically divides an object into 16 KB segments. All segments of the same object are assumed to have the same popularity. This is not unrealistic in P2P file-sharing systems, which unlike video streaming systems download segments in random order and all segments are needed for the object to be useful. Therefore, for clarity of the presentation, we treat segments as small objects. Given the first assumption, each cache can store up to k segments. Our third assumption in the analysis is that caches observe similar relative popularity of objects. That is, popular objects in one AS are likely to be popular in other ASs. This is also not unrealistic in P2P systems, which have no sense of network locality and in which popular objects typically attract global client populations. Our traces confirm this intuition as discussed in Section 7.4.

7.2 Performance of Cooperative Caching

The goal of our analysis is to determine the saving achieved due to cooperation. To do this, we first compute the cost in the independent caching case. Then, we compute the cost in the cooperative caching case and compare them.

For the independent caching case, no cooperation among caches is performed, and we assume that each cache uses a popularity-based local replacement policy such as the least frequently used (LFU) policy. Given the above assumptions, the cost of serving N objects from clients in an AS is given by

$$C_{ind} = \tau_l \sum_{i=1}^{N} f(i) + \tau_e \sum_{i=k+1}^{N} f(i), \tag{2}$$

where $f(i)$ is the probability of accessing the object at rank i, and τ_l and τ_e are the costs of downloading an object from the local cache and an external source, respectively. Note that objects are ranked based on their relative popularity such that $f(i) \geq f(j)$ for all $i < j$. The above equation represents the average cost because, for large N, in the steady state, the cache stores the most k popular objects. The second term in (2) is the cost of retrieving objects $k+1, k+2, \ldots, N$ from external sources because the first k objects are stored in the cache. The first term is the cost of all objects because they impose local τ_l cost regardless whether they are stored in the cache or not. Note also that because of the lack of cooperation and by the similar relative popularity of objects assumption, all caches will end up storing the same top k popular objects.

In the cooperative caching case, when a cache receives a request for an object that it does not store, it first forwards this request to other caches in the cache group. If any of them has the requested object, the object is served to the client (with a cost $\tau_g + \tau_l$). If otherwise, the object is downloaded from an external source (with a cost $\tau_e + \tau_l$). In addition, the caches coordinate the replacement of objects. In particular, when an object eviction needs to be made by a cache, that cache chooses the least popular object in the cache group. The cost of serving N objects from clients in an AS in the cooperative caching case has three components: cost of serving locally stored objects, cost of serving objects stored on other caches in the cache group, and cost of serving objects from external sources. Thus, the total cost is given by

$$C_{coop} = \tau_l \sum_{i=1}^{N} f(i) + \tau_g \frac{m-1}{m} \sum_{i=1}^{mk} f(i) + \tau_e \sum_{i=mk+1}^{N} f(i). \tag{3}$$

Note that the middle term in (3) represents the additional objects stored in the cache group. Since $\tau_g \leq \tau_e$, the cooperative caching model will keep at most one copy of any object in the cache group. Thus, the total number of objects stored in the cache group is mk. Consider any cache in the cache group. If this cache receives a request for any of the mk objects stored in the cache group, it will serve it locally with probability $1/m$, and from another cache in the group with probability $(m-1)/m$. In the latter case, there is an additional cost of τ_g to serve the object.

We define the relative saving in cost due to the cooperation as $\Psi = (C_{ind} - C_{coop})/C_{ind}$. After substituting the Mandelbrot-Zipf popularity model in (2) and (3), we get the following:

$$\Psi = \frac{\tau_e \sum_{i=k+1}^{mk} \frac{1}{(i+q)^\alpha} - \tau_g \frac{m-1}{m} \sum_{i=1}^{mk} \frac{1}{(i+q)^\alpha}}{\tau_l \sum_{i=1}^{n} \frac{1}{(i+q)^\alpha} + \tau_e \sum_{i=k+1}^{n} \frac{1}{(i+q)^\alpha}}. \tag{4}$$

To simplify the above equation, we set the local cost τ_l to 0 because it is typically much smaller than the external cost τ_e and the intercache cost τ_g. In addition, τ_l is incurred in both the independent and the cooperative caching cases. Thus, it is not a differentiating factor in determining the potential gain of cooperation. We further define $\tau = \tau_g/\tau_e$, which is the relative cost of serving an object from the cache group to the cost of serving it from an external source. We carry out our analysis in terms of τ. We also denote the expression $\sum_{i=x}^{y} \frac{1}{(i+q)^\alpha}$ by $S(x, y)$ for clarity. Thus, the saving in the cost due to the cooperation among m caches is given by

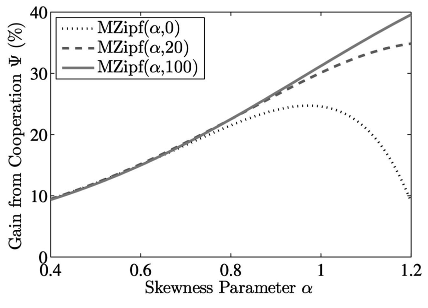
$$\Psi = \frac{1}{S(k+1, N)} \left[S(k+1, mk) - \tau \frac{m-1}{m} S(1, mk) \right]. \tag{5}$$

The gain from cooperative caching in the above equation models the three most important aspects of the system: 1) traffic characteristics in terms of α and q of the Mandelbrot-Zipf popularity distribution; 2) network characteristics captured by τ; and 3) the cache group characteristics captured by the number of caches m and the capacity of each cache k.

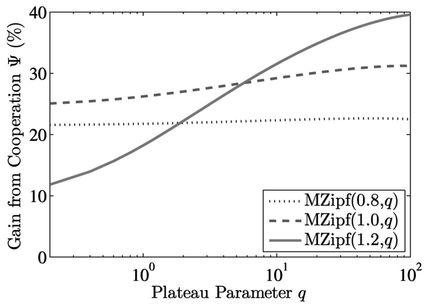
7.3 Numerical Analysis

We numerically analyze the gain (or saving in the cost) due to the cooperative caching given by (5). We consider one parameter at a time, while fixing all others at reasonable values. We start by studying the impact of traffic characteristics on the gain from cooperative caching. In our analysis, the traffic characteristics are modeled by the skewness parameter α and the plateau parameter q of the Mandelbrot-Zipf popularity distribution. We fix the number of caches m at 8 and τ at 0.1. We also fix the storage capacity of individual caches. The storage capacity is represented as the ratio of objects that can be stored in the cache k to the total number of objects N. Note that N is the number of unique objects, not the total amount of P2P traffic. We call k/N the relative cache size, and it is set to 0.5 percent.

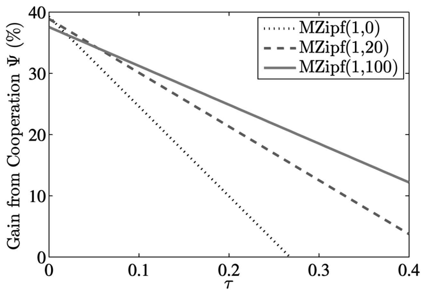
In Fig. 10a, we vary α between 0.4 and 1.2. Larger α values mean that the popularity curve is more skewed, which implies that the top-ranked objects receive higher



(a)



(b)



(c)

Fig. 10. Numerical results of the analytic model. (a) and (b) The impact of traffic characteristics on the gain from cooperative caching. (c) The impact of changing the relative cost of internal to external links τ on the gain from cooperative caching.

fractions of the requests. We plot the gain from cooperation Ψ versus α for three representative q values: 0, 20, and 100. As shown in the figure, a significant gain of up to 40 percent can be achieved by cooperative caching. Fig. 10a demonstrates an interesting feature in cooperative caching: the gain is more robust against larger values of the skewness parameter α for P2P traffic than for Web traffic. Web traffic follows a Zipf-like distribution [20], which is a Mandelbrot-Zipf with $q = 0$. Thus, as shown in Fig. 10a, when α increases, the gain from cooperation diminishes quickly for Web traffic. On the other hand, the plateau factor q somewhat mitigates the effect of large α values for P2P traffic. This is because the Mandelbrot-Zipf distribution has a flattened head (see Fig. 3), which indicates that even with large α values, the requests are spread over more objects in the head of the popularity distribution. Since more objects in the head of the popularity distribution require larger storage capacities, cache cooperation will be more beneficial.

Next, we analyze the impact of changing the plateau parameter q. In Fig. 10b, we vary q between 0 and 100, and plot the gain from cooperation Ψ for three values of α. The results in Fig. 10b indicate that large values of q achieve higher gains from cooperation. This is because when q is small, the head of the popularity distribution is less flattened and the impact of the skewness parameter on the gain is higher. Nonetheless, for typical q values (more than 10 as shown in [13]), the gain from cooperation is at least 20 percent.

Next, we vary τ and study the gain from cooperation. Fig. 10c shows that the gain from cooperation is still achievable even if the relative cost of serving an object from the cache group to the cost of serving it from external sources is fairly large. This is more apparent when the plateau parameter is not zero, which is the typical case in P2P traffic. This confirms the viability of cooperative caching for P2P traffic in different environments. Finally, we analyze the impact of the cache group characteristics on the gain from cooperative caching. We vary the number of caches from 2 to 24 and compute the gain from cooperation. The results, shown in Fig. 11, indicate that the cooperation is beneficial for cache groups of different sizes.

7.4 Analysis Validation Using Simulation

In this section, we relax the assumptions made in the analysis and verify that our results still hold. We simulate a cooperative caching group that uses the online replacement policies: iLFU and cLFU. Our simulator uses synthetic traces with controlled (realistic) object and popularity characteristics, which are generated as follows: We create 10^5 objects and we choose their sizes based on the object size distribution observed in our traces collected from the widely deployed Gnutella P2P system. The popularity for these objects is assigned using the Mandelbrot-Zipf distribution with different values for α and q. We use eight caches in the simulation. A total of 10^6 requests are generated for all caches. We also randomly assign τ values for the caches, i.e., the cost of downloading an object from another cache is no longer a constant. As in the analytical analysis, we define the gain from cooperation Ψ, as the total cost of iLFU minus cLFU normalized by iLFU.

We study the impact of all parameters on the gain Ψ, including α, q, cache size, and τ. Samples of our results are shown in Figs. 12a and 12b. In Fig. 12a, we vary the parameters of the popularity distribution and measure the achieved gain. The figure shows very similar pattern as Fig. 10a: the gain in cooperative caching is robust against wide ranges of α and q values. The figure also shows that for $q = 0$ (i.e., Web traffic), the gain drops quickly as the skewness parameter α increases. In Fig. 12b, we vary the network parameter τ. Again, simulation results are similar to the analytical results in Fig. 10c. The results confirm that in P2P systems where q is large, the gain from cooperation is significant for different network environments. Note that our goal from this simulation is to validate the behavior (not the exact performance values) of cooperative caching predicted by our analytic analysis. We used different (more

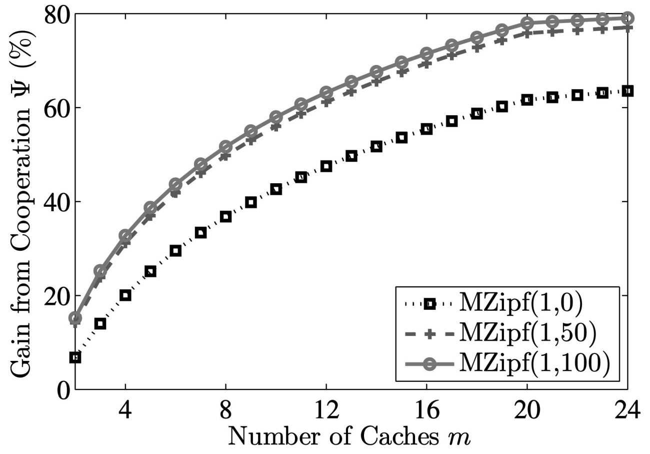
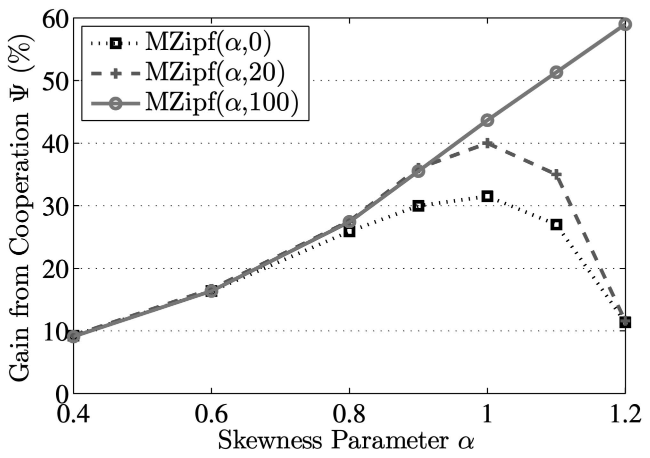
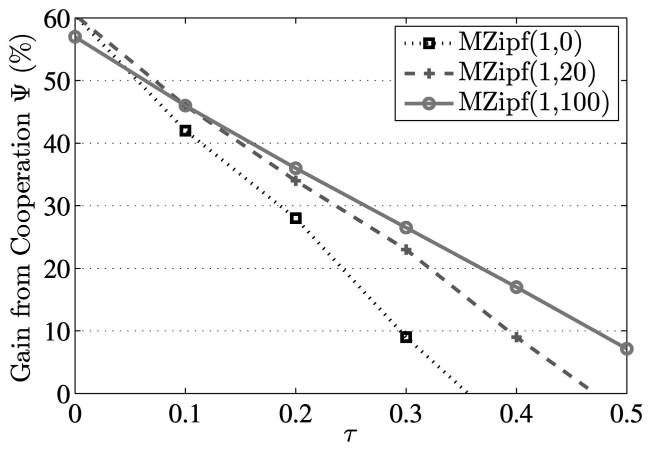


Fig. 11. The impact of number of caches on the gain from cooperation.



(a)



(b)

Fig. 12. Validation of our analytic results. (a) and (b) Simulation experiments using online policies on actual traces yield similar behavior as our analytic model.

realistic) parameters in the simulation than the ones used in the numerical analysis in Section 7.3.

Finally, we verify that the assumption of similar object popularity in different ASs is realistic. We study the popularity of objects in different ASs from our Gnutella traces. We choose the most popular 100 objects across all ASs, i.e., objects that received the highest number of requests in all ASs combined (global popularity). Then, we find the most popular 100 objects in each AS (local per-AS popularity). We compute for each AS the percentage of objects that are common in the top 100 globally popular objects and the top 100 locally popular objects. The results are shown in Fig. 13 for 10 different ASs (the dark bars). The figure shows that for any of the 10 ASs, at least about 30 percent of the locally popular objects are also globally popular. The figure also compares the top 100 locally popular objects versus the top 200 globally popular objects (the light bars). The results show that a significant portion of the locally popular objects observes high popularity in all ASs. We repeated the experiment for the top 1,000 and 2,000 popular objects and obtained similar results.

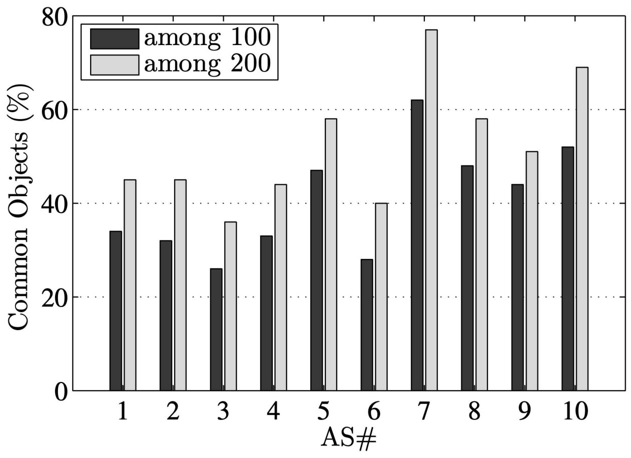


Fig. 13. Validating the assumption that objects in P2P systems observe somewhat similar popularity in different ASs.

8 Conclusions

In this paper, we analyzed the potential gain of cooperative caching for P2P traffic. We proposed two models for cooperation: 1) among caches deployed in different ASs and 2) among caches deployed within a large AS. In both models, caches cooperate to save bandwidth on expensive WAN links. We collected traces from an eight-month measurement study on a popular P2P system. The traces describe object requests that would have been seen by many caches if they were deployed in ASs operating in different geographical regions and have different number of clients. We designed many trace-based simulation experiments to rigorously analyze various aspects of cooperative caching. Our results show that cooperative caching is viable for P2P traffic because it could improve the byte hit rate by up to 330 percent in some cases. Considering the huge volume of the P2P traffic, even 1 percent improvement in byte hit rate accounts to saving in the order of terabytes of traffic on the expensive WAN links. The large savings from cooperation could serve as an incentive for ISPs to deploy caches and enable cooperation among them. Our results also show that the overhead imposed because of cooperation among caches is negligible, less than 0.003 percent of the total traffic.

In addition, we proposed simple models for object replacement policies in cooperative caching systems. These models allow an individual cache to cooperate with other caches, but without harming its own performance. This is done by making the decision to replace an object from the cache based only on local information from that cache. Furthermore, we used an analytic model to assess the gain from cooperative caching under different traffic and network characteristics. Our model confirms that substantial gains from cooperation are possible under wide ranges of traffic and network characteristics. We validated the results from our analysis using simulations, where most of the assumptions made in the analysis were relaxed.

Acknowledgments

This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," *Proc. ACM Symp. Operating Systems Principles (SOSP '03),* pp. 314-329, Oct. 2003.

[2] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *IEEE/ACM Trans. Networking,* vol. 12, no. 2, pp. 219-232, Apr. 2004.

[3] T. Karagiannis, A. Broido, N. Brownlee, K.C. Claffy, and M. Faloutsos, "Is P2P Dying or Just Hiding?" *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '04),* pp. 1532-1538, Nov. 2004.

[4] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet Service Providers Fear Peer-Assisted Content Distribution?" *Proc. ACM Conf. Internet Measurement (IMC '05),* pp. 63-76, Oct. 2005.

[5] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06),* pp. 1-9, July, 2006.

[6] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak, "Cache Replacement Policies Revisited: The Case of P2P Traffic," *Proc. Int'l Workshop Global and Peer-to-Peer Computing (GP2P '04),* pp. 182-189, Apr. 2004.

[7] Packeteer Web Page, http://www.packeteer.com/, 2008.

[8] P-Cube IP Service Control Web Page, http://www.p-cube.net/indexold.shtml, 2008.

[9] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit, "Are File Swapping Networks Cacheable? Characterizing P2P Traffic," *Proc. Int'l Workshop Web Content Caching and Distribution (WCW '02),* Aug. 2002.

[10] Home Page of CacheLogic, http://www.cachelogic.com/, 2009.

[11] Home Page of PeerCache, http://www.joltid.com/, 2009.

[12] Home Page of Sandvine, http://www.sandvine.com/, 2009.

[13] M. Hefeeda and O. Saleh, "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems," *IEEE/ACM Trans. Networking,* vol. 16, no. 6, pp. 1447-1460, Dec. 2008.

[14] S. Dykes and K. Robbins, "Limitations and Benefits of Cooperative Proxy Caching," *IEEE J. Selected Areas Comm.,* vol. 20, no. 7, pp. 1290-1304, Sept. 2002.

[15] K.-W. Lee, K. Amiri, S. Sahu, and C. Venkatramani, "Understanding the Potential Benefits of Cooperation among Proxies: Taxonomy and Analysis," IBM Research Report RC22173, Sept. 2001.

[16] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the Scale and Performance of Cooperative Web Proxy Caching," *Proc. ACM Symp. Operating Systems Principles (SOSP '99),* Dec. 1999.

[17] O. Saleh and M. Hefeeda, "Modeling and Caching of Peer-to-Peer Traffic," *Proc. IEEE Int'l Conf. Network Protocols (ICNP '06),* pp. 249-258, Nov. 2006.

[18] M. Hefeeda, C. Hsu, and K. Mokhtarian, "pCache: A Proxy Cache for Peer-to-Peer Traffic," *Proc. ACM SIGCOMM,* Aug. 2008.

[19] G. Shen, Y. Wang, Y. Xiong, B. Zhao, and Z. Zhang, "HPTP: Relieving the Tension between ISPs and P2P," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS '07),* Feb. 2007.

[20] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM,* pp. 126-134, Mar. 1999.

[21] A. Klemm, C. Lindemann, M.K. Vernon, and O.P. Waldhorst, "Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems," *Proc. ACM Internet Measurement Conf. (IMC '04),* pp. 55-67, Oct. 2004.

[22] Squid Web Proxy Cache Home Page, http://www.squid-cache.org/, 2009.

[23] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures," *Proc. Int'l World Wide Web Conf. (WWW '04),* pp. 512-521, May, 2004.

[24] T. Karagiannis, M. Faloutsos, and K. Claffy, "Transport Layer Identification of P2P Traffic," *Proc. ACM Internet Measurement Conf. (IMC '04),* pp. 121-134, Oct. 2004.

[25] S. Gadde, J. Chase, and M. Rabinovich, "A Taste of Crispy Squid," *Proc. Workshop Internet Server Performance (WISP),* 1998.

[26] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches," *Proc. Workshop Hot Topics in Operating Systems,* pp. 93-98, May, 1997.

[27] D. Wessels and K. Claffy, *Internet Cache Protocol (ICP), Version 2,* RFC 2186, Sept. 1997.

[28] D. Wessels and K. Claffy, "ICP and the Squid Web Cache," *IEEE J. Selected Areas Comm.,* vol. 16, no. 3, pp. 345-357, 1998.

[29] X. Dmitiropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. Claffy, and G. Riley, "AS Relationships: Inference and Validation," *ACM SIGCOMM Computer Comm. Rev.,* vol. 37, no. 1, pp. 31-40, Jan. 2007.

[30] W. Norton, "The Evolution of the US Internet Peering Ecosystem," http://www.nanog.org/mtg-0405/norton.html, white paper, Nov. 2003.

[31] Gnutella Home Page, http://www.gnutella.com, 2008.

[32] Limewire Home Page, http://www.limewire.com/, 2009.

[33] Z. Silagadze, "Citations and the Zipf-Mandelbrot's Law," *Complex Systems,* vol. 11, pp. 487-499, 1997.

[34] GeoIP Database Home Page, http://www.maxmind.com, 2009.

[35] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM Trans. Networking,* vol. 12, no. 1, pp. 1-14, Feb. 2004.

[36] "Web Page of Rocketfuel: An ISP Topology Mapping Engine," http://www.cs.washington.edu/research/networking/rocketfuel/, 2009.

Mohamed Hefeeda (S'01, M'04, SM'09) received the BSc and MSc degrees from Mansoura University, Egypt, in 1994 and 1997, respectively, and the PhD degree from Purdue University, West Lafayette, Indiana, in 2004. He is an assistant professor in the School of Computing Science, Simon Fraser University, Surrey, British Columbia, Canada, where he leads the Network Systems Lab. His research interests include multimedia networking over wired and wireless networks, peer-to-peer systems, network security, and wireless sensor networks. Mohamed has served as the program chair of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2010) and as the vice chair of the Distributed Multimedia Track in the International Conference on Embedded and Multimedia Computing (EMC 2010). He has served on many technical program committees of major conferences in his research areas, including ACM Multimedia, ACM Multimedia Systems, ACM/SPIE Multimedia Computing and Networking (MMCN), IEEE Conference on Network Protocols (ICNP), and IEEE Conference on Communications (ICC). He is on the editorial boards of the *Journal of Multimedia* and the *International Journal of Advanced Media and Communication*. He served as the guest editor of the special issue on high-quality multimedia streaming in P2P environments of the *International Journal of Advanced Media and Communication*. His paper on the hardness of optimally broadcasting multiple video streams with different bit rates won the Best Paper Award in the IEEE Innovations 2008 conference. In addition to publications, he and his students develop actual systems, such as PROMISE, pCache, svcAuth, pCDN, and mobile TV testbed, and contribute the source code to the research community. The mobile TV testbed software developed by his group won the Best Technical Demo Award in the ACM Multimedia 2008 conference. He is a senior member of the IEEE and a member of the ACM SIGCOMM and SIGMM.

Behrooz Noorizadeh received the BSc degree in computer engineering from Sharif University of Technology, Iran, in 2005, and the MSc degree in computing science from Simon Fraser University, British Columbia, Canada, in 2007. He is currently a software engineer at Nokia, Vancouver, British Columbia, Canada. His research interests include peer-to-peer systems, Internet caching algorithms, and distributed networks.