Cooperative Caching: The Case for P2P Traffic

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

Behrooz Noorizadeh
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

***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 (ASes), 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 actual traffic traces for multiple caches. Then, we perform an extensive trace-based simulation study to analyze different angles of cooperative caching schemes. Our results demonstrate that: (i) significant improvement in byte hit rate can be achieved using cooperative caching, (ii) simple object replacement policies are sufficient to achieve that gain, and (iii) the overhead imposed by cooperative caching is negligible.**

I. Introduction

Peer-to-peer (P2P) systems currently generate a major fraction of the total Internet traffic [1], accounting for as much as 60–70% of the traffic in some Internet service providers (ISPs). Previous studies, e.g., [2], have shown that the huge volume of the P2P traffic has negative consequences on ISPs, because it multiplies the load on their backbone links and increases the possibilities of network congestion. Several approaches have been proposed in the literature to reduce the negative impacts of P2P traffic. These include enhancing traffic locality [2] and traffic caching [3]. More aggressive approaches using devices for traffic blocking and shaping have also been used in practice [4]. 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 to further reduce the amount of traffic downloaded from sources outside of the local network domain.

We focus on exploring the full potential of P2P traffic caching. 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 [5]. In addition, caching does not require changing 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, including CacheLogic [6], PeerCache [7], and Sandvine [8]. Efficient caching algorithms have also been proposed in the literature [3], [9]. 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 [10], and even in these cases the gain may not be significant [11].

In this paper, we propose and rigorously analyze two models for cooperative caching of P2P traffic. The first model enables cooperation among caches that belong to different autonomous systems (ASes), 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 actual traffic traces for multiple caches. Then, we perform an extensive trace-based simulation study to analyze different angles of cooperative caching schemes. Our results demonstrate that: (i) significant improvement in byte hit rate can be achieved using cooperative caching, (ii) simple object replacement policies are sufficient to achieve that gain, and (iii) the overhead imposed by cooperative caching is negligible.

The rest of this paper is organized as follows. In Sec. II, we summarize the related work. In Sec. III, we describe the proposed two models for cooperative caching. In Sec. IV, we present our measurement study and the methods we use to construct traces for different caches. In Sec. V, we present extensive trace-based simulation experiments to show the potential of cooperative caching. In Sec. VI, we propose and analyze several object replacement policies for cooperative caching. We also analyze the overhead introduced because of cooperation among caches. We conclude the paper in Sec. VII.

II. Related Work

The benefits of caching P2P traffic have been shown in [5]. The authors show that P2P traffic is highly repetitive and therefore amenable to caching. The study in [2] suggests deploying caches or making P2P protocols locality-aware to reduce the load on ISP networks. No caching algorithms

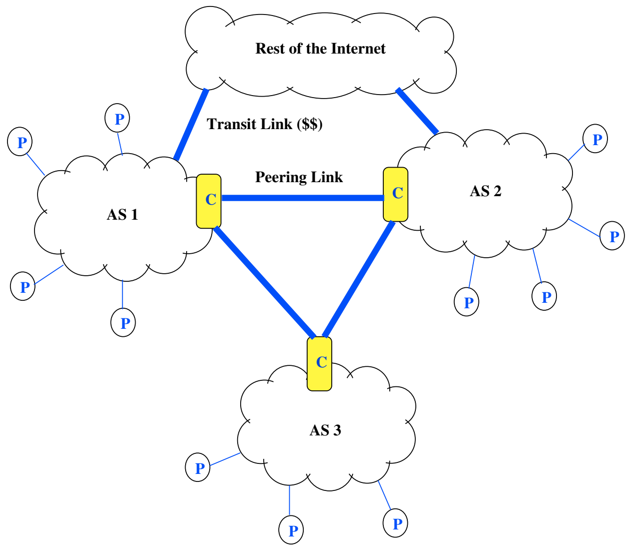


Fig. 1. Cooperation among caches in different Ases.

were proposed in [2], [5] . Caching algorithms designed for P2P traffic have been proposed in [3] and [9]. All the of above works target independent caches and do not consider cooperation among caches to further enhance the byte hit rate.

Cooperative caching for web traffic has been extensively studied, see for example [10]–[12] and the references therein. Using trace-based simulation and analytical analysis, the authors of [11] argue that cooperation yields marginal benefits for web caching. The gain from cooperation in web caching is debatable because: (i) web objects are fairly dynamic, (ii) a web proxy cache may be able to store most of the popular objects locally, (iii) the overhead imposed is high relative to object sizes, and (iv) 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], [9], 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 hundreds milliseconds of latency to a P2P download is not a critical concern, because many sessions take long periods (minutes and even hours) and they 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.

III. Models for Caching P2P Traffic

In independent caching, a cache is to be deployed near the gateway routers of autonomous systems (ASes) that choose to employ caching to reduce the burden of P2P traffic. As argued in [3], [9], 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 is defined as the ratio of the number of bytes served from the cache to the total number of bytes transfered. Notice 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 transfered—may not be well defined in the P2P case [3]. This is because requests in P2P systems are typically issued for *segments* of objects, not for entire objects. Thus the byte hit rate is more suitable than the hit rate for studying caching of P2P traffic.

We present the two proposed models for cooperation among caches in the following subsections.

A. Cooperative Proxy Caches in Different ASes

The first model for cooperation considered in this paper is depicted in Fig. 1. In this model, caches deployed in different ASes cooperate with each other to serve requests from clients in their networks. The cooperating ASes 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 bandwidth on long-haul, inter-city, 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 ASes are interconnected through a very high-speed (Gb/s optical links) network called BCNET. The bandwidth among university ASes is abundant. On the other hand, the university ASes 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 Mb/s) 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 look up inside the cache group can be done in several ways. For example, a centralized directory can be used, similar to the CRISP protocol [13] proposed for cooperative web caching. The look up 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 look up methods. One distributed look up method is using the Internet Cache Protocol (ICP) [14]. 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 issue requests for segments of objects.

B. *Cooperative Proxy Caches within the same AS*

The second model for cooperation proposed in this paper is for caches deployed within the same AS. 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) inter-connected with high-speed optical links. ISPs provide Internet access to their customers at POPs. The links inside an ISP are usually over provisioned. 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). Deploying cooperative caches in such large ISP 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 [15]. Caching would also benefit clients because their traffic will traverse fewer inter-ISP links, which are more susceptible to overload and congestion.

As a concrete example for this model of cooperative caching, we show in Fig. 2 the distribution of clients in a large AS in the US: AS 1859 (AT&T-Comecast). We discuss how we created this map in Sec. V. 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 from crossing the boundary of the AS and consuming bandwidth on expensive inter-ISP links. The cooperation among these caches employs protocols similar to the ones described in the previous subsection.

We note that cooperation among caches within the same AS would be easier to implement in practice than cooperation among caches in different ASes. This is because in the former case all caches are owned and managed by a single entity, while in the latter 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 ASes to enable cooperative caching. Finally, we should 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 are outside the scope of this paper.

IV. 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 [6]–[8], we are not aware of any public traces that can be used to study caching of P2P traffic.

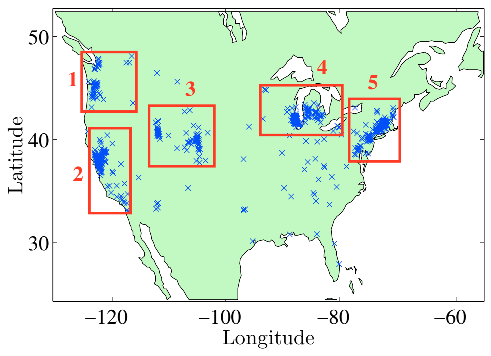


Fig. 2. Locations of peers in a large AS in the US (AS 1859–AT&T-Comcast)

We conducted a passive measurement study of the Gnutella file-sharing network, which is one of the top-three most popular P2P systems. Gnutella has two kinds of peers: ultra peers, characterized by high bandwidth and long connection periods, and leaf peers which are ordinary peers that only connect to ultra peers. Peers exchange several types of messages including PING, PONG, QUERY and QUERYHIT. A QUERY message contains search keywords, a TTL field and the address of the immediate neighbor which forwarded the message to the current peer. Query messages are propagated to all neighbors in the overlay up to a hop distance specified by the TTL field. If a peer has one or more of the requested files, it replies with a QUERYHIT message. A QUERYHIT message is routed on the reverse path of the QUERY message it is responding to, and it contains the URN (uniform resource name) of the file, IP address of the responding peer, and file size.

We modified a popular Gnutella client, called Limewire, to run as a monitoring node. We ran our monitoring node in the ultra-peer 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 ultra peer, we configured it to reach most of the Gnutella network as follows. We increased the number of concurrent connections that it could maintain to be up to 500. A regular ultra peer allows up to 16 connections to other ultra peers and a maximum of 30 to leaf peers. Effectively, our peer is worth more than 20–30 regular ultra peers. In many times, our peer was concurrently connected to more than 350 other ultra peers. Let us assume that each of these 350 ultra peers connect to other 10 ultra peers on average, each of them connect to other 10, and so on. Given that queries in Gnutella are forwarded up to several (typically 7) hops among ultra peers, our monitoring node was able to capture traffic from a huge number of peers. In addition, our monitoring node ran continuously for eight months, while other peers joined and

TABLE I
SUMMARY STATISTICS FOR THE P2P TRAFFIC OBSERVED IN 10 ASES IN THE WEST COAST OF NORTH AMERICA.

AS#	AS Name	Unique objects (TB)	BHR, 0.5TB cache (%)	BHR, 1TB cache (%)
2161	AT&T	30.41	11.0	15.1
9548	Road Runner	14.37	9.3	15.4
9406	Verizon	12.05	13.4	20.4
11394	Charter	14.99	9.0	14.7
11715	Cox	12.43	12.3	19.1
1859	AT&T-Comcast	59.67	3.8	6.5
1782	Shaw	27.96	6.7	11.0
233	Telus	19.02	8.4	13.7
18952	Comcast	17.51	9.3	14.2
105	Qwest	14.02	8.9	15.1

left the network. This means that the 200—400 other peers connected to our node were continuously changing, which allowed it to reach different and larger portions of the Gnutella network. It is important to emphasize that our monitoring node captured traffic from numerous ASes, not only our local AS. This is because of its high connectivity to many other ultra-peers. During the eight months of the study, 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 ASes. The total amount of traffic observed was more than 6,000 tera bytes.

We construct the traces for individual ASes as follows. For a given AS, we use unique QUERYHIT messages to count the number of replicas of each object found in that AS. We divide QUERYHIT messages among ASes based on the source IP addresses contained in them. We use the GeoIP database [16] in mapping IPs to ASes. Most object replicas found in an AS were downloaded sometime in the past, and a cache would have seen a sequence of requests for these objects if it had been deployed in that AS. This assumes that most of the downloads were supplied by peers from outside the AS, which is actually the case because peers in most current P2P networks have no sense of network proximity and thus do not favor local peers over non-local peers. The measurement study in [1] has shown that up to 86% of the *locally-available* objects were downloaded from external peers. Thus, we construct the sequence of requests from the unique QUERYHIT messages, i.e., the sequence has one request for each replica downloaded by a peer. Peers who replied earlier with QUERYHITs for an object are assumed to have downloaded the object earlier. Notice 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. In addition, various P2P traffic characteristics extracted from these sequences in different ASes are similar to those reported in previous studies. Therefore, because of the long duration, huge amount of traffic recorded, and matching results with previous works, we believe that our traces are representative to what would be observed by caches deployed in different ASes.

V. THE POTENTIAL OF COOPERATIVE CACHING

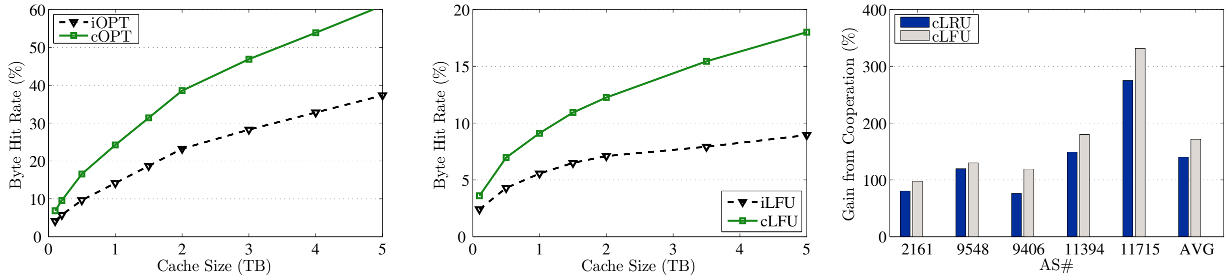
In this section, we use our traces to study various aspects of cooperative caching. We start by showing that cooperative caching is needed to achieve high byte hit rates and to save bandwidth on expensive links. Then, we demonstrate the potential gain from cooperation in the two models proposed in this paper. We study the gain under offline object replacement policies to show the upper bounds on the gain as well as under realistic online replacement policies. We also analyze the relative gain from cooperation achieved by different ASes.

A. The Need for Cooperation

We start our analysis by making the case for cooperative caching of P2P traffic. We choose 10 different ASes 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 ASes with the largest amount of traffic seen in our traces to make our results statistically significant. For each of these 10 ASes, 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 [16]. 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 I lists the names and summary statistics for these 10 ASes.

As Table I shows, the total size of *unique* objects observed in each AS is too large to fit in a single cache. Note that the total amount of traffic in each AS is much larger than the total size of unique objects, because the former accounts for the number of times each object is requested. Notice also that these statistics are for the data our monitoring node was able to capture from only one P2P system (Gnutella). Therefore, the actual amount of P2P data for each AS is much larger and indeed the unique objects cannot fit into one cache. In addition, the probability of accessing objects in P2P systems is not concentrated in a few objects [1], [9]. 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.

To confirm the above intuition, we simulate an independent cache for each of the 10 ASes. The cache uses an *optimal* offline algorithm denoted by iOPT. iOPT looks at the entire trace offline and stores the most popular objects that can fill the cache. This simulation gives an upper bound on the achievable byte hit rates with independent caches. We run the simulation for two cache sizes: 0.5 TB and 1.0 TB. The achieved byte hit rates are shown in the fourth and fifth columns of Table I. As shown in the table, the optimal byte hit rate achievable with independent caches is less than 10% with a 0.5 TB cache in most cases, and it is slightly improved for a 1 TB cache.



(a) cOPT vs. iOPT (b) cLFU vs. iLFU (c) Normalized gain

Fig. 3. Comparison between independent and cooperative caching for five caches deployed in five ASes in LA.

This is a fairly small *theoretical bound* for the byte hit rate, and the practical online replacement algorithms will have even smaller byte hit rates.

Therefore, given that individual caches do 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 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 subsections, and what costs are involved, which we analyze in Sec. VI-B.

B. *Gain from Cooperation among Caches in Different ASes*

We consider cooperation among ASes within the same city. We select two large cities: New York City (NYC) and Los Angeles (LA). In each city, we choose five popular ASes 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 ASes are created using the same IP-to-AS and IP-to-geolocation mapping methods explained in Sec. V-A.

To assess the full potential of cooperative caching, we compute the optimal achievable byte hit rate when the caches cooperate *perfectly* with each other. By perfect cooperation we mean that if an object exists in one of the caches it will definitely be found by any other cache looking for it. We denote this cooperation as cOPT. cOPT looks at the entire traces of all caches and stores the most popular objects in them without exceeding their storage capacities. cOPT does not store an object in more than one cache, because it assumes that the cost of retrieving objects from another cache in the group is smaller than the cost of retrieving objects from outside the group. cOPT represents the upper bound on the byte hit rate in the cooperative caching case.

We conduct several simulation experiments to compare the upper bound on the byte hit rate in the cooperative case (resulted by using cOPT) versus the corresponding upper bound in the independent caching case (resulted by using iOPT). We analyze the performance of the two cache groups in LA and NYC. We vary the size of individual caches from 100 GB to 5 TB and compute the byte hit rate achieved by iOPT and cOPT for each cache in the two cache groups. Fig. 3(a) shows a sample of the results for AS 2161 in LA; other ASes achieved similar results. The figure implies that significant gains in byte hit rate can be achieved by cooperation among caches. For example, with a cache of size 1 TB, the byte hit rate of cOPT is about 24%, which is almost double of iOPT. In addition, the gain improves as the cache size increases, which is expected in the future as storage prices keep dropping.

The above results are obtained using the theoretical cOPT and iOPT offline policies to study the bounds on the gain. We conduct another set of experiments to study the gain under two realistic (online) replacement policies: cLRU (cooperative Least Recently Used) and cLFU (cooperative Least Frequently Used). 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. The setup in this set of experiments is the same as in the previous one, except we compare cLFU versus iLFU and cLRU versus iLRU, instead of comparing cOPT versus iOPT. A sample of the results is shown in Fig. 3(b). The figure indicates that substantial improvements in byte hit rates can also be achieved using realistic policies.

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. 3(c); the results for the cache group in NYC are similar. The figure shows that up to 330% improvement in byte hit rate can be achieved for some ASes. The average gain across all five ASes is more than 120%. Fig. 3(c) also indicates that different replacement policies may yield different gains and, more importantly, that some ASes may benefit more than others from cooperation. We elaborate more on this important issue in Sec. V-D and Sec. VI-A.

The experiments in this subsection show that the byte hit rate could be doubled or tripled in some cases because of cooperation. Considering the huge volume of the P2P traffic, even 1% improvement in byte hit rate accounts to saving in the order of tera bytes 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.

C. *Gain from Cooperation among Caches within the same AS*

Next, we study the potential gain from cooperation between caches deployed within the same AS. We choose several large ASes 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 a sample AS: AS 1859 (AT&T-Comecast), which is a large ISP in the US. As mentioned in Sec. III-B, the points of presence (POPs) of an AS are the usual locations for deploying caches. The exact locations of ISP's POPs, however, are not public information. 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. Notice 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 cOPT, cLFU, and cLRU policies versus iOPT, iLFU, and iLRU policies, respectively. The results (figures not shown due to space limitations) 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: when multiple caches are deployed within the same AS, and when caches are deployed in different ASes in the same geographical area.

D. *Relative Gain from Cooperation by Different ASes*

We noticed in Sec. V-B that some ASes may benefit more than others from cooperation. The experiments in that section are limited to only five ASes and they are all fairly large in terms of the amount of traffic seen in each AS. In this subsection, we expand the level of cooperation to include 64 ASes with different sizes and we study the relative gain of each AS. For each of these 64 ASes, 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 ASes based on the number of objects that are requested in their corresponding trace files.

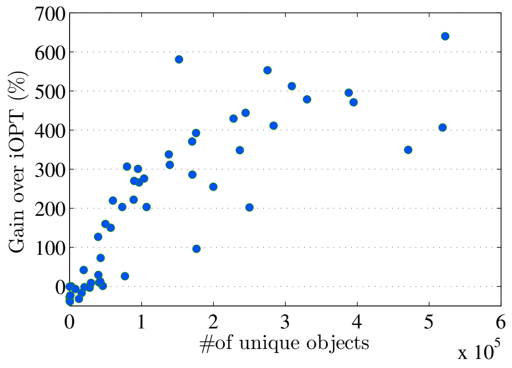


Fig. 4. Relative gain in byte hit rates when 64 ASes of different sizes cooperate with each other using cOPT.

We simulate a cooperative cache group that contains all of the 64 ASes. The object replacement policies used are iOPT and cOPT. We measure the byte hit rate achieved by each AS when it cooperates with the others and uses cOPT. We also measure the byte hit rates when ASes do not cooperate with each other, but each of them deploys a cache and uses iOPT. We compute the gain in byte hit rate due to cooperation (cOPT - iOPT) and normalize it by the byte hit rate of iOPT. We summarize the results in the scatter diagram in Fig. 4, 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. The figure indicates that while some ASes improve their byte hit rates by up to 600%, others achieve much less gain from the cooperation, and several ASes actually have *negative* gain. The figure also shows that the small ASes (in terms of amount of traffic) are mostly the victim. This is because small ASes may replace their own popular objects (based on requests from their own clients) by globally more popular objects. Recall that cOPT maximizes the byte hit rate across all caches, not individual ones. This unequal (and sometimes negative) gain may discourage some ASes to cooperate. We show in Sec. VI that careful design of replacement policies eliminates the possibility of negative gains and shrinks the gap in the gain achieved by different ASes without sacrificing the total byte hit rate.

VI. Replacement Policies and 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

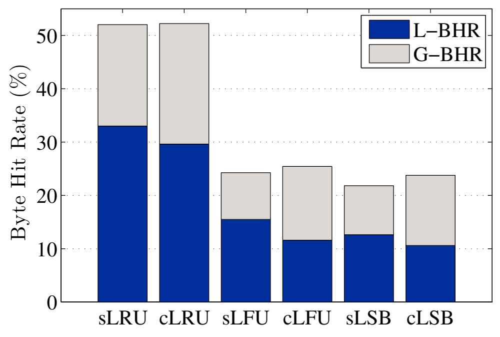


Fig. 5. Comparison between six replacement policies in terms of achieved group (G-BHR) and local (L-BHR) byte hit rates.

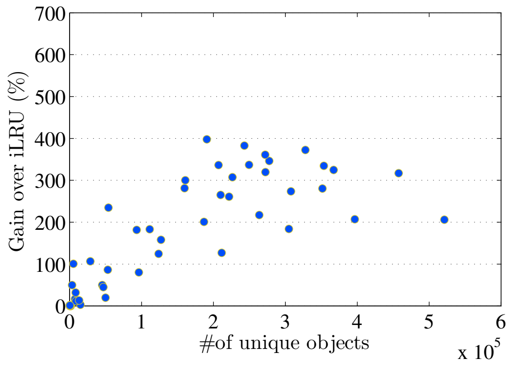


Fig. 6. Relative gain in byte hit rates when 64 ASes of different sizes cooperate with each other using sLRU.

the cache group, as we will demonstrate in this section. A balanced relative gain by all participating ASes is critical to the success of the cooperative caching for P2P traffic. 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 evaluate the overhead imposed due to cooperation among caches.

A. *Replacement Policies for Cooperative Caching*

We have already described two online replacement policies for cooperative caching in Sec. V-B: cLFU and cLRU. cLFU and cLRU implement group-wide LFU and LRU policies, respectively. The authors of [3] have proposed a few replacement policies designed for caching P2P traffic. The Least-Sent Byte (LSB) was shown to outperform others in [3]. 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 ASes 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: (i) two for cooperation among caches in different ASes (the five caches in LA and the five caches in NYC), and (ii) 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 8-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 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 refers 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. 5. The figure shows the achieved byte hit rates (L-BHR and G-BHR) by two individual ASes as well as the average over all five ASes in LA. Similar results were obtained for the other cache groups in NYC, AS 11715, and AS 1859. The results shown in Fig. 5 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 [9], as popular objects tend to stay popular for some time, then they gradually lose popularity.

Replacement Policies and Relative Gain from Cooperation. We study the effect of the replacement policies on the gain achieved by different ASes. As mentioned in Sec. V-D, our goal is to find the policies that eliminate negative gains incurred by some ASes and reduce the gap between percentage of improvements in byte hit rate between ASes with various sizes. To study this effect, we repeat the experiment described in Sec. V-D, which simulates the cooperation among 64 ASes with different sizes. The experiment is repeated several times and each time a different replacement policy is used and the gain in byte hit rate of each AS is computed. To facilitate visual comparisons, we represent the results of each experiment as a scatter diagram with the same scales for the x-axis and y-axis. Our results, a sample of them is shown in Fig. 6, imply that: (i) the selfish replacement policies sLRU, sLFU, and sLSB eliminate the possibility of negative gains, and (ii) among the selfish policies, sLRU produces the smallest improvement gap between ASes. For example, comparing Fig. 6 versus Fig. 4 indicates that there are a fewer number of ASes that achieve gain less than 10% under sLRU than under cOPT. Moreover, most of the dots representing gains of different ASes for cOPT in Fig. 4 are spread over a larger range of the y-axis than they are for sLRU in Fig. 6.

B. Overhead Analysis in Cooperative Caching

By overhead we mean the additional number of bytes transmitted beyond the transfer of the requested objects themselves. As mentioned in Sec. III-A, we use the Internet Cache Protocol (ICP) [14] to facilitate communication and object look up 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 transfered bytes. The results (figures not shown) imply that the maximum overhead imposed by cooperative caching is less than 0.003% for all policies, which is indeed negligible. The results also show that sLRU has the smallest overhead. This is because sLRU has higher local byte hit rate (L-BHR), as discussed in the previous subsection. Local hits do not impose overhead, because they do not require sending ICP queries to other caches.

VII. Conclusions

In this paper, we considered the potential gain of cooperative caching for P2P traffic. We proposed two models for cooperation: (i) among caches deployed in different ASes, and (ii) 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 ASes 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% in some cases. Considering the huge volume of the P2P traffic, even 1% improvement in byte hit rate accounts to saving in the order of tera bytes 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 the overhead imposed because of cooperation among caches is negligible, less than 0.003% 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. We showed that the proposed replacement policies not only eliminate the possibility that some ASes suffer negative gains from cooperation, but they also shrink the gap in the gain achieved by ASes with different sizes. This is achieved without sacrificing the total byte hit rate.

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," in *Proc. of ACM Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, NY, October 2003, pp. 314–329.

[2] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in *Proc. of ACM Conference on Internet Measurement (IMC'05)*, Berkeley, CA, October 2005, pp. 63–76.

[3] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak, "Cache replacement policies revisited: The case of P2P traffic," in *Proc. of International Workshop on Global and Peer-to-Peer Computing (GP2P'04)*, Chicago, IL, April 2004, pp. 182–189.

[4] "Packeteer web page," http://www.packeteer.com/.

[5] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit, "Are file swapping networks cacheable? Characterizing P2P traffic," in *Proc. of International Workshop on Web Content Caching and Distribution (WCW'02)*, Boulder, CO, August 2002.

[6] "Home Page of CacheLogic," http://www.cachelogic.com/.

[7] "Home Page of PeerCache," http://www.joltid.com/.

[8] "Home Page of Sandvine," http://www.sandvine.com/.

[9] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Transactions on Networking*, 2007, accepted to Appear.

[10] S. Dykes and K. Robbins, "Limitations and benefits of cooperative proxy caching," *IEEE Journal of Selected Areas in Communications*, vol. 20, no. 7, pp. 1290–1304, September 2002.

[11] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative web proxy caching," in *Proc. of ACM Symposium on Operating Systems Principles (SOSP'99)*, Kiawah Island, SC, December 1999.

[12] K.-W. Lee, K. Amiri, S. Sahu, and C. Venkatramani, "Understanding the potential benefits of cooperation among proxies: Taxonomy and analysis," *RC22173, IBM Research Report*, September 2001.

[13] S. Gadde, J. Chase, and M. Rabinovich, "A Taste of Crispy Squid," in *Proc. of Workshop on Internet Server Performance (WISP'98)*, 1998.

[14] D. Wessels and K. Claffy, "ICP and the Squid Web cache," *IEEE Journal on Selected Areas in Communication*, vol. 16, no. 3, pp. 345–357, 1998.

[15] W. Norton, "The evolution of the U.S. Internet peering ecosystem," http://www.nanog.org/mtg-0405/norton.html, November 2003, white Paper.

[16] "GeoIP Database Home Page," http://www.maxmind.com.