Network-Aware Forward Caching

Jeffrey Erman, Alexandre Gerber, Mohammad T. Hajiaghayi, Dan Pei, Oliver Spatscheck
AT&T Labs Research
New Jersey, USA
erman,gerber,hajiagha,peidan,spatsch}@research.att.com

ABSTRACT

This paper proposes and evaluates a Network Aware Forward Caching approach for determining the optimal deployment strategy of forward caches to a network. A key advantage of this approach is that we can reduce the network costs associated with forward caching to maximize the benefit obtained from their deployment. We show in our simulation that a 37% increase to net benefits could be achieved over the standard method of full cache deployment to cache all POPs traffic. In addition, we show that this maximal point occurs when only 68% of the total traffic is cached.

Another contribution of this paper is the analysis we use to motivate and evaluate this problem. We characterize the Internet traffic of 100K subscribers of a US residential broadband provider. We use both layer 4 and layer 7 analysis to investigate the traffic volumes of the flows as well as study the general characteristics of the applications used. We show that HTTP is a dominant protocol and account for 68% of the total downstream traffic and that 34% of that traffic is multimedia. In addition, we show that multimedia content using HTTP exhibits a 83% annualized growth rate and other HTTP traffic has a 53% growth rate versus the 26% over all annual growth rate of broadband traffic. This shows that HTTP traffic will become ever more dominent and increase the potential caching opportunities. Furthermore, we characterize the core backbone traffic of this broadband provider to measure the distance travelled by content and traffic. We find that CDN traffic is much more efficient than P2P content and that there is large skew in the Air Miles between POP in a typical network. Our findings show that there are many opportunties in broadband provider networks to optimize how traffic is delivered and cached.

Categories and Subject Descriptors

D.4.8 [**Performance**]: Measurements—*web caching*

General Terms

Networking Optimization

1. INTRODUCTION

Over the past decade, as the new “killer" Internet applications emerge, new content delivery mechanisms are invented to meet the demand of these new applications. This is evidenced by the following two examples. Web (text/image) is the first “killer" Internet application, thus HTTP protocol dominated Internet traffic usage in the first several years of the Internet [10, 29]. Web caches and Content Distribution Networks (CDNs) were thus invented to improve Web performance. When Peer-to-Peer (P2P) file sharing became popular a few years later, it was shown to be dominant in studies conducted in 2002 to 2005 for DSL, cable, and fiber broadband networks [7, 15, 22]. A lot of approaches have been proposed to improve the delivery efficiency of P2P. With the recent emergence of user-generated video, social networking, and TV/Movie-on-demand services, most of which run on top of HTTP, we suspect that HTTP protocol has become the dominant content delivery protocol, especially for multimedia (video/audio) content. For example, an informal report in a *lightreading* article in 2006 [1] indicates that “network operators are reporting a rise in overall web traffic and a rise in HTTP video streaming". Furthermore, the multimedia content we see today might be just the tip of the iceberg of what is coming in the next decade as more providers join the business and make more, higher-definition content available and more subscribers access it. As such, we believe it is timely and important to investigate the new opportunities for more efficient delivery mechanisms for HTTP traffic again.

A natural solution is forward caching (proposed in the 90s), which is to deploy HTTP caches within an Internet Service Provider’s (ISP) network caching *all* cacheable HTTP traffic accessed by the customers of the ISP. Caching makes intuitive sense in that Internet content popularity is often very skewed thus offering good opportunity for reusing. This is especially true for video content, as shown in recent studies [5, 6, 16]. Unfortunately, most large US based ISPs currently do not operate forward caches within their network. The main reason for this decision lies in the economics of deploying forward caches. Forward caches are additional hardware components (typically UNIX servers) which have to be purchased, deployed and managed at a large number of locations. In the US the bandwidth savings can often not justify the cost of such a server deployment. To reduce the costs associated with forward caching in an ISP, we believe that the most cost effective way of deploying forward caches is to only deploy them in selected POPs (Point of Presences) caching only selected *expensive-to-deliver* content. We call this approach Network Aware Forward Caching.

To motivate and justify our solution, we first provide a systematic measurement study of the characteristics of the content transmitted over the Internet today and shows the dominance of HTTP traffic, followed by the comparison of the efficiency of existing delivery mechanisms (HTTP, CDN and P2P), and the cacheability of HTTP content. We then formulate the cache placement problem, and propose and evaluate our heuristics.

To show how dominant HTTP content is compared to other types of content transmitted over the Internet today, we characterize the Internet traffic of 100K subscribers of a US residential broadband provider. Using both layer 4 and layer 7 analysis, we confirm our

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

hypothesis that HTTP is the dominant protocol, which contributes to 68% of the total downstream peak traffic. Furthermore, HTTP has become the workhorse for data delivery: 80% of multimedia content uses HTTP as its distribution protocol (mainly flash video) and file downloads via HTTP contributes 10% of downstream traffic in contrast to 0.3% for FTP. This is a drastic change since a few years ago, when HTTP contributed to 9% of the total traffic [22], P2P was dominiant [7, 15, 22], and 63% of total residential volume was user-to-user traffic [7]. Our results show that recently the volume of HTTP content per subscriber is increasing much faster than the 26% overall increase rate: multimedia streams over HTTP and other HTTP traffic exhibits a 83% and a 53% annualized growth rate, respectively. This means that HTTP's share will even increase further.

Our study shows that among the existing delivery mechanisms, CDNs are currently serving already a significant portion (46%) of the large file transfers in the network and are efficient in bringing the content closer to the content consumers, much better than the existing P2P technologies or typical HTTP content providers by a factor of 2 to 3 when comparing average bit-distance. Distance travelled on the network is strongly related to the network cost of the traffic. Furthermore, the distance traversed on a network by different sources of traffic to different points of presence (POPs) varies significantly. The traffic to some remote POPs can benefit significantly from better delivery mechanisms such as forward caching. In the HTTP traffic cacheability study, we found that 60% of the traffic can potentially be reused overall because it is requested more than once. However, only 33% of the content is suited for an optimized delivery infrastructure when adding more realistic constraints.

Finally, based on these observations, we introduce and evaluate our new proposal, network aware forward caching, that increases efficiency, reduces backbone traffic and network costs and increases end-user performance. Contrary to a simple all-or-nothing forward caching deployment in a network, we argue that, by being network aware, partial deployments of forward caches for a subset of the POPs and a subset of the traffic sources provides greater benefits per dollar invested. Indeed, we just showed the disparity in efficiency and the fact that some sources, such as CDNs are already efficient and, therefore, don't need to be cached. In addition to that observation about the differences in efficiency of each source to each POP, we also note that an Internet Service Provider incurs different costs based on the nature of the neighbor sending the traffic (e.g. transit traffic is more expensive than peering traffic, and customer traffic even generates revenue), and we include this additional dimension in our decision process when selecting the content that will be stored on the caches. We formulate the problem of determining the optimal deployment, and shows that it is NP-hard. We propose one pseudo-polynomial algorithm that solves the exact problem, and a greedy heuristic algorithm that is much faster. Our evaluation of the heuristic algorithm using realistic data shows that the optimal deployment in terms of network costs occurs when only 68% of the total traffic is cached. Moreover, that solution is clearly senstive to the backbone costs and the caching costs.

The remainder of the paper is organized as follows. Section 2 describes our measurement methodology. Section 3 presents over-all content composition results. Section 4 studies how current delivery mechanisms work, and Section 5 presents the content cachability results. Section 6 presents the formulation, algorithms, and evaluation results for our network aware forward caching approach. Section 7 reviews related work, and Section 8 concludes the paper.

2. MEASUREMENT METHODOLOGY

This section presents our network monitoring infrastructure and data sets used in the analysis of this paper. To achieve our objectives in this paper of characterizing broadband traffic usage and evaluating our network aware caching approach, we obtained traces from multiple vantage points of a US Broadband Providers network to understand the different aspects concerning the delivery of content to a subscriber. In particular, the three types of data sets that we utilize in our analysis are as follows: aggregated traffic records from broadband subscriber aggregation access points, aggregated netflow records of the core backbone traffic, and unsampled HTTP header records from a single aggregation point. In the following subsections we elaborate on each type data set. Our analysis is based on a US Provider and networks in different geographic locations may not exhibit the same characteristics.

2.1 Access Traffic

Our network monitoring infrastructure [11] consists of five network monitors analyzing traffic from $100K$ subscribers of a US broadband provider. The monitors are diversely located on five BRAS (Broadband Remote Access Servers; an aggregation point of Digital Subscribers Lines, or DSLs) in three states (California, Texas and Illinois). The subscriber data we analyzed was from the week of February 25, 2007 to September 30, 2008.

For our study, network monitors summarized the observed traffic volumes every 5 minutes into flow records. The flow records measure the number of packets and bytes for each identified applications class (described below). To reduce resource consumption, the network monitors perform a combination of packet sampling and flow sampling when computing these flow records [12]. The 5-minute flow records are used to compute hourly summaries of the aggregate traffic from all the subscribers observed at the monitor.

Our application classification relies on application header, heuristic, and port number analysis to determine a final classification of a flow into an application class.

Overall, we classify our traffic into 16 categories: Web (HTTP), Multimedia (HTTP, RTSP, RMTP, etc), File Sharing (P2P), Games (Quake, WoW), Net News (NNTP), Business (VPN, Databases), VoIP (SIP), Chat (IM, IRC), Mail (POP3, IMAP), DNS, FTP, Security Threats, Other, ICMP, Other-TCP and Other-UDP. (The examples in brackets are non-exhaustive of what we identify.) We base these categories on a determination of the use of the data and not explicitly on the protocol. This is most significant for HTTP traffic, which we classified into either Web or Multimedia category. Therefore, we separate HTTP traffic by mime type–if the mime type of a flow is for a video or audio format we classify this flow as HTTP Multimedia instead of HTTP Web. In addition, Gnutella and BitTorrent tracker-based HTTP traffic is classified as those specific P2P applications.

We adopt many of the same packet payload signatures described by Sen *et al.* [25] and Karagiannis *et al.* [20]. We also use additional signatures to identify application subclasses. For instance, as outlined above we use the mime type information in HTTP flows to further classify HTTP flows. In addition, we extract from the control channel the information needed to identify the data channel of the FTP, RTSP, and Skype protocols.

Additional P2P traffic was identified with other P2P specific heuristics. For example, we use the announced port in the tracker messages for BitTorrent to identify incoming encrypted flows as P2P. However, due to P2P applications such as BitTorrent using encryption to obfuscate their protocols, we believe much of our unknown TCP traffic (in the Other-TCP class) is P2P as well. We have based this inference on additional analysis we performed on

the Other-TCP traffic. In particular, we have found that based on the flow characteristics and the payload information in the flows this traffic is consistent with encrypted P2P traffic. Generating and validation of additional signatures for encrypted P2P traffic is left as future work.

Lastly, we use *some* layer 4 port numbers to identify any remaining traffic not classified using signatures and heuristics.

2.2 Core Backbone Traffic

Another type of data utilized in our analysis are aggregated NetFlow [9] records from the core backbone of the network. This data allows us to measure the ingress and egress traffic on the various Peering, Transit or On-Net links of the network. The data used in our analysis is from September 27, 2008 until October 5, 2008. The data sets were obtained using a NetFlow collected on the backbone router of a US Broadband Provider. These traces provide the amount of ingress and egress traffic volume between core routers. To minimize the performance impact on the routers, we used deterministic packet sampling with a rate of 1 packet out of 500. We then use smart sampling [12] to further reduce the data volume.

2.3 HTTP Traffic

To facilitate the study of HTTP cacheability, we analyzed the HTTP header data from approximately 20K subscribers during the week of January 27, 2008 to February 2, 2008. During our analysis we correlated the HTTP requests with the actual TCP flows to obtain the actual flow sizes. This step was taken because not all content sizes are available in the `Content-Length` field of the HTTP header. For instance, some objects such as dynamically generated pages or streaming content will sometimes have a Content-Length of 0. In our analysis we used from the HTTP header fields the GET, POST, Cache-Control, Pragma, and Content-Length fields of the HTTP header and the IP address of the web server. All other information such as the subscriber IP address and cookie values were not analyzed. In total, we analyzed approximately 550 million requests representing 45 TB of traffic with an average request size of 91 KB. Unlike the smart-sampled records in the previous two types of data sets we use, the HTTP data in our analysis is unsampled and includes all requests.

3. BROADBAND TRAFFIC OVERVIEW

This section provides an overview of the growth and overall application breakdown in the broadband data we studied. We show that the volume and fraction of multimedia content delivered using HTTP is increasing rapidly and that P2P traffic as a percentage of traffic is actually decreasing. HTTP is more generally becoming the protocol of choice for various kinds of activities such as software distribution, multimedia and P2P applications.

3.1 Growth of Broadband Traffic

Figure 1 shows that a large US broadband ISP saw a relatively stable growth rate of the average downstream traffic per subscriber of 26% per year, with an above average growth rate for the last 2 years. An interesting observation when looking closer at the busy hour traffic per subscriber is that, while in the first years of this period, the average downstream traffic per subscriber was growing faster than the downstream traffic per subscriber, the opposite can now be observed in the last 2 years. This indicates that synchronous applications (e.g. instant watching multimedia streams) are recently growing faster than asynchronous applications (e.g. P2P file sharing). It is also important to keep in mind that the behavior during the busy hour is what really matters. The Internet infrastructure is engineered for the peak demand and that generally

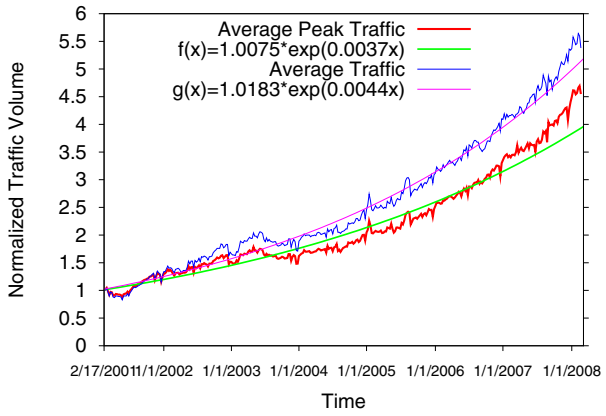

Figure 1: Weekly average downstream traffic per subscriber during the busy hour based on several million subscribers

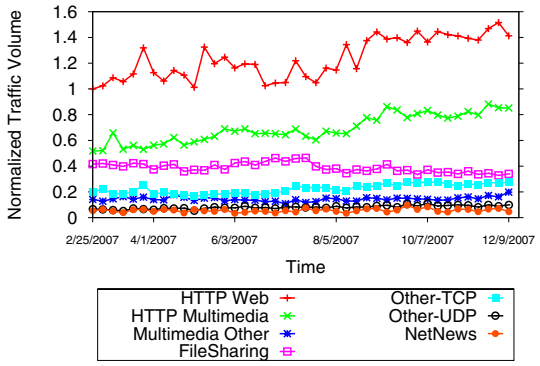

Figure 2: Normalized Weekly traffic per Application Class during the Busy Hour

has spare capacity the rest of the time. The busy hour is defined as the 1-hour time span during a day which exhibits the largest average traffic.

Figure 2 shows the weekly normalized application traffic volumes during the busy hour based on our aggregate BRAS traffic records. There are two major trends that can be seen. The first is that HTTP traffic, both Web and Multimedia, shows consistent growth during the busy hour over the observed period. In the figure, HTTP Multimedia and HTTP Web traffic exhibit a 83.1% and a 52.9% annualized growth rate, respectively. The second is that P2P traffic has remained steady and shows a decline in its percentage share of the overall traffic mix. These may be important observations as they contradict reports claiming that P2P traffic is continuing to increase at dramatic rates. The growth of HTTP traffic and especially the HTTP Multimedia traffic is the most significant cause of broadband subscribers traffic growth.

3.2 Broadband Subscriber Traffic Mix

Next, we look at the characterization of the overall application mixture of the traffic. Table 1 shows the application mixture of the downstream and upstream traffic at all five monitors during the week of January 28, 2008 to February 3, 2008. The downstream traffic volumes are typically 3 times greater than the upstream traffic volumes during the busy hour. HTTP is the dominant protocol and accounts for the largest volume of traffic on the network and

Table 1: Application Mix During the Busy Hour

Class	Downstream		Upstream	
	Busy Hour	Average	Busy Hour	Average
HTTP Web	41.8%	39.8%	23%	17.5%
HTTP Multimedia	25.8%	21.5%	2.7%	1.9%
FileSharing	9%	12.3%	25%	27.6%
Other-TCP	8.2%	9.7%	26.2%	30.4%
Multimedia Other	4.7%	4.7%	2.1%	1.7%
Other-UDP	2.5%	3%	10.3%	10.2%
Games	1.4%	1.3%	3.2%	2.3%
NetNews	1.4%	1.6%	0.1%	0.1%
Business	1.1%	1.7%	2.2%	2.8%
Voip	1%	1.1%	1.1%	0.9%
Chat	0.6%	0.5%	0.5%	0.4%
Mail	0.5%	0.8%	0.9%	1.2%
Dns	0.5%	0.4%	0.6%	0.4%
Ftp	0.3%	0.3%	0.4%	0.3%
SecurityThreat	0.2%	0.4%	0.5%	0.7%
Other	0%	0%	0%	0.1%
ICMP	0%	0%	0.3%	0.3%

Table 2: HTTP Breakdown

Class	% Busy Hour Traffic	% Average Traffic
/http/video	30%	31.9%
/http/text-image	25.6%	25.9%
/http/download	18.9%	16.2%
/http/javascript	5.6%	5.8%
/http/audio	4.4%	5.5%
/http/other	4.4%	4.4%
/http/flash	4.1%	4.4%
/http/https	3.7%	2.8%
/http/otherapp	1.5%	1.2%
/http/xml	1%	1.1%
/http/binary	0.2%	0%
/http/office	0.1%	0.1%
/http/rss	0%	0%

accounts for 68% of the downstream traffic during the busy hour. HTTP has also taken over file downloads and FTP is now only a very small percentage of the overall traffic.

P2P makes up at least 9% of the downstream traffic during the busy hour and 12.3% of all downstream traffic. If we assume in part that much of the TCP-Other traffic is due to encrypted or unidentified P2P traffic, then P2P still makes up a maximum of 17% of the downstream traffic. As we have already noted, P2P's percentage share of the overall traffic has been decreasing. However, P2P is still the dominant protocol in the upstream. Because P2P protocols are designed to exploit subcribers' sharing of data, this class of traffic has more symmetrical data flows than Web/Multimedia protocols that are generally asymmetric [2]. The volume of P2P traffic has been quoted with various statistics in the media [4] and literature [19]. In many cases, the value quoted is the upstream volume. Depending on the time and direction we could state a number between 17% to 58%. In a DSL environment, upstream link capacity is dedicated per subscriber and shared backbone links have symmetric capacity. Therefore, upstream traffic is not a problem and the traffic in the busiest direction in the busy hour is more of interest (i.e., the downstream direction). However, for cable-based ISPs, the P2P upstream volume may be more of a factor as subscribers share the capacity of local cable lines.

Network News accounts for a surprising amount of the traffic. This is due to NetNews being used to download large files such as movies, music, and software.

We turn next to the categories of HTTP and Multimedia to get a better understanding of specific protocols and applications that are used.

Table 2 shows the breakdown of the subcategories for HTTP traffic (HTTP Web in Table 1). When a file is requested using the HTTP protocol, the HTTP header in the servers response includes a `Content-Type` field which contains the mime type of the file. We have based these subcategories primarily on the mime types extracted from the HTTP stream. For example, an image file might have the mime type of `image/gif` and would be classified as /web/text-image. In our categories, we group similar mime types together.

Table 3: Multimedia Breakdown (includes HTTP Multimedia)

Class	% Busy Hour Traffic	% Average Traffic
/http/video	70.1%	70.9%
/http/audio	10.4%	12.3%
/multimedia/rtmp	7.9%	7.3%
/multimedia/rtsp	6.6%	5.5%
/multimedia/rtp	3%	2.5%
/multimedia/shoutcast	0.8%	0.5%
/multimedia/ms-streaming	0.5%	0.4%
/multimedia/other	0.2%	0.1%
/multimedia/realaudio	0.1%	0.1%
/multimedia/h323	0%	0%

The /web/no-http subcategory is not based on mime type. This category is for traffic on a standard HTTP port that does not use the HTTP protocol. This unidentified traffic makes up 7.1% of the web traffic. Though there is no traffic shaping to be evaded on the broadband network being studied, this could be the result of encrypted P2P applications using a default HTTP port to avoid firewalls and traffic shaping on other ISP networks.

An interesting category is the /web/download. This traffic is for the download of compressed (e.g., .zip, .rar., .tar.gz) or executable files (e.g., .exe). This shows that HTTP has replaced FTP as the distribution mechanism for these types of files and software patches. HTTP download traffic accounts for 10.2% of all traffic, whereas, the volume of FTP traffic is a quite small, 0.3%, as shown in Table 1.

Table 3 shows the breakdown of Multimedia traffic (HTTP Multimedia in Table 1) into subcategories. HTTP is used to provide more than 80% of the multimedia data, substantially more than the 20% of multimedia traffic supported by traditional multimedia streaming protocols such as RTSP and RTMP. Upon further investigation, we found that 85% of the /web/video is flash video (flv) used by popular User Generated Content (UGC) sites like YouTube to deliver video content. These UGC sites account for about 40% of total multimedia traffic. Some of the possible reasons why HTTP is used to provide so much of the multimedia content on the Internet could be the result of no license fee costs for streaming servers (as required by traditional multimedia streaming protocols), compatability between operating systems, ability to easily traverse firewalls, and ease of integration into CDN services.

The conclusions we draw from this section are that HTTP is the prevalent protocol on the Internet, accounting for 66% of the traffic and is the main driver of per subscriber broadband traffic grow today. It is very much the workhorse for data delivery and is very versatile. HTTP has moved beyond its historical role in delivery of web text and image content and is increasingly being used to handle most of the Internet's tasks, such as distribution of software, updates, patches, and multimedia, and by P2P applications (gnutella, torrent trackers, torrent distribution).

4. DELIVERY MECHANISMS

4.1 Efficiency of Content Distribution

An important consideration for the performance of a data distribution mechanisms is the distance the data travels to reach the broadband subscriber. The distance traveled directly affects the efficiency of data delivery, and minimizing it is of interest to both the broadband subscriber and the ISP. Average bit-distance traveled is strongly related to the direct network costs associated with transfer-

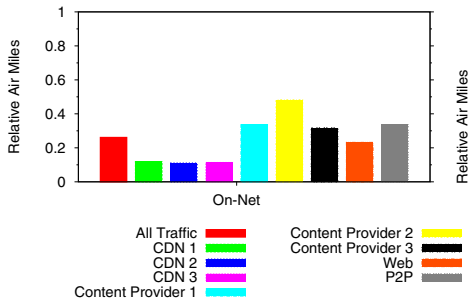

Figure 3: Air Miles for Different Content Providers

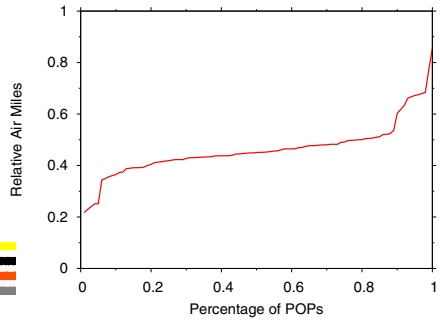

Figure 4: Average Air Miles Per POP

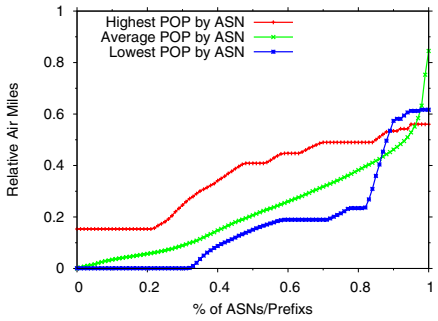

Figure 5: Air Miles By ASN/Prefix

ring the data. We conducted this part of our study using NetFlow data from the backbone of a Tier-1 US ISP.

From the perspective of the broadband subscriber, increased travel distances affects the load time of web pages and file downloads, and reduces throughput (e.g., TCP throughput is limited by the round trip time). One method content providers use to enhance the quality of their data delivery is to outsource it to a CDN to place the data objects closer to the users, for fast, efficient access. From the perspective of the ISP, the network miles data travels reflects the direct cost of delivery of the data over their backbone, so shorter distances mean lower costs.

We can calculate the efficiency and speed of data transfer by measuring the average bit-distance traveled assuming direct connectivity, which we call *Air Miles*. We calculate the distance traveled as the direct physical distance between two end points. Thus, Air Miles can be calculated as: AM = sum of distance each bit travels / total number of bits. In order to remove the impact of interdomain routing, we isolate the traffic exchanged between customers (On-Net traffic), following intradomain routing from the source to the destination. We would like to note that the best metric to use would be layer 1 route miles; however, computation of this is difficult and average bit-distance metric is a close approximate of route miles.

The data we used for our analysis is from September 27, 2008 until October 5, 2008. Note that for this data we only used L4 application mappings to obtain application information.

Figure 3 shows ON-Net air miles of different anonymized content providers, CDNs, P2P, and Web. The *ALL traffic* represents all the traffic including P2P and Web. Overall, the distance traveled by P2P applications are typically 25% longer than the distance of HTTP traffic. Currently, the CDN's air miles are 2 to 3 times lower than P2P and other content providers such as large web sites. This indicates that CDNs are effective and are having a significant impact on the delivery of data.

There are several conclusions we can draw from this section. The first is that the current generation of P2P applications are quite inefficient in air miles. However, there have been some progress recently to address this issue with P2P. For instance, the P4P Working Group has been working on P4P (Proactive network Provider Participation for P2P) to use topology information from ISPs to optimize P2P traffic on P2P networks. Xie *et al.* show where broadband subscribers experience increased throughput using P2P applications due to significantly more data being served On-Net [8, 31]. The second conclusion is that CDNs are doing a good job of bringing data closer to the end user. This allows the users to have a better multimedia experience because these large multimedia files can be served from a CDN and obtain higher bitrates. This allows more and better quality content to be consumed and show that large files are typically served from CDNs.

4.2 Air Mile Differences Between POPs

Figure 4 shows the distribution of relative air miles travelled to POPs (Point of Presence) in the network. Figure 5 shows for the POPs with the highest, lowest, and an average of all POPs the distribution of air miles to different ASNs. The main observation to take away from graphs is that on the Broadband Providers network there are many opportunities for optimizations. The traffic going to some POPs are significantly more expensive, on average, to transport than others. In addition, at each POP there are some ASN that are significantly more expensive, on average, to transport as well. These observations help to motivate our proposed caching solution later presented in Section 6.

5. CACHEABILITY

Many applications on the Internet have data flows that could potentially be reused to save network bandwidth because the same content is being requested more than once. For instance, multiple requests to the same web page could be cached and either served from a local or network cache. Another example is multiple users streaming the same video file, which could be served using multicast or P2P.

The application classes of Web, Multimedia, P2P, and Network News are the most likely candidates for transferring content that is reusable. These traffic classes were shown in Section 3.1 to represent 89% of the network traffic during the busy hour. Other traffic classes such as VoIP, Chat and Business are less unlikely to contain any reusable content in their data flows but represent only 11% of the network traffic. The analysis in this section is based off of our HTTP traffic trace. By using these HTTP records we can look at how much traffic is reusable for Web and Multimedia because 95% application classes are served using the HTTP protocol.

5.1 Time-To-Live Analysis

The Time-To-Live (TTL) length specified in `Control-Cache` field of the HTTP records indicates how long an object should be kept in a cache before it needs to be refreshed.

Figure 6 shows the CDF of the TTL length specified by different file sizes. We find noticeable amounts of records with TTLs set as: 1 hour, 1 day, and 7 days.

We also looked at the impact CDNs have on the temporal characteristics of the traffic. To identify which requests are from a CDN we employ a similar methodology as Huang *et al.* [17]. To facilitate this analysis we used dumps of the DNS tables used by the subscribers at the broadband ISP. These dumps were collected on February 2nd, 2008.

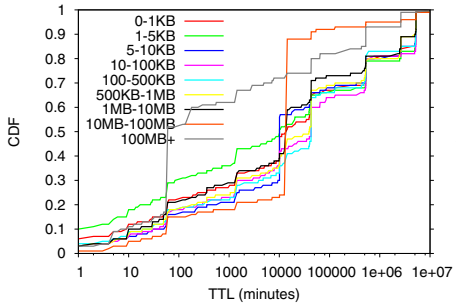

Figure 6: TTL of Cache Directives for HTTP Requests

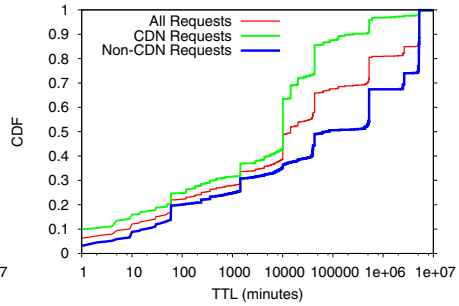

Figure 7: TTL of Cache Directives for Requests

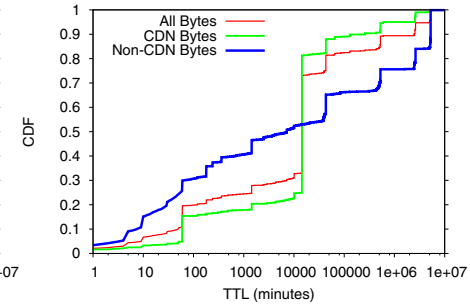

Figure 8: TTL of Cache Directives for Weighted by Bytes

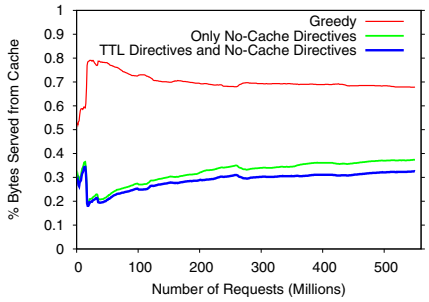

Figure 9: Webcache Results

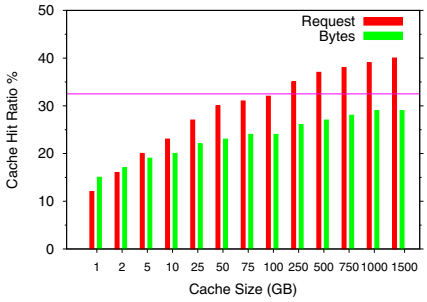

Figure 10: Caching Results using Different Cache Sizes

Table 4: Percentage of Bytes for Large Flow vs. Small Flows of CDNs and Other

Class	<1MB	1-10MB	10-100MB	100MB +	Overall
CDN	8.0%	7.0%	18.2%	46.1%	8.0%
Other	92.0%	93.0%	81.8%	53.9%	92.0%
Total	100.0%	100.0%	100.0%	100.0%	100.0%

We use a two-fold approach to identify CDN requests. The first step in our analysis is to look up the DNS entry for each hostname in the HTTP requests. If the hostname resolves to a CNAME that belongs to a CDN provider then we label these requests as CDN traffic. In the second step, for the remaining traffic we use the server IP address to resolve the AS number. If the AS number is from a known CDN provider then we label these requests as CDN traffic as well.

Figure 7 and Figure 8 show the CDFs of the TTL lengths when weighted by requests and total bytes, respectively. When comparing both graphs together we observed that while the distribution for requests and bytes are similar for small TTLs. The TTLs for bytes is smaller indicating that large file sizes have larger TTLs on average than smaller file sizes. A noticeable artifact in the graph is that for the byte distribution of CDN requests over 50% of the bytes have a TTL of 10 days. However, overall we found that there were no other substantial differences in how the TTLs were set between CDN and non-CDN traffic that would indicate a difference in the cachability of the CDN content.

Table 4 shows the percentage of all bytes in the downstream traffic broken down by source. We found that a significant amount of the large files are being served by CDNs. In particular, over 46% of flows greater than 100 Mbytes in size are originating from a CDN today.

5.2 Cache Analysis

Not all HTTP requests are cacheable. This occurs for a variety of reasons–the web page may contain private information like a cookie, or is dynamically generated. In the HTTP 1.1 protocol there are two fields, `Cache-Control` and `Pragma`, in the HTTP header that can be used to indicate cacheability [14]. The `Content-Control` field can be used by a server to indicate if the document is not cacheable or the time the document can be kept before it is stale. The `Pragma` field can be used by the client to indicate a request for a fresh copy of the file. For instance, hitting the refresh button on most browsers causes the HTTP request to be marked as `no-cache` when sent. We used both of these fields in the HTTP records we analyzed.

If we remove from the HTTP traffic the 42% that is marked non-cacheable, in total, approximately 60% of all traffic is potentially reusable. We define content as reusable if there is more than one request for a specific object during the period the object is valid (e.g. not stale or modified). The results in Figure 9 assume that the cache has an unlimited size. Thus, we do not assume removal of items from the cache.

These assumptions are more general in some cases and provide a potentially more optimistic result than if explicitly studying caching. Our methodology of identifying uncacheable documents and using an unlimited cache size is similar to the approach taken by Feldmann *et al.* who completed a comprehensive studies of HTTP headers in relation to caching in 1999 [13].

We have tested three different algorithms for calculating reusability. The first, which we denote as "Greedy", is where we ignore the Content-control and Pragma directives including TTLs. In our second algorithm, denoted as "No-Cache", we only respect the directives that indicate the content should not be cached or not served from cache. The third algorithm, denoted as "TTL-Cache", takes into account all directives and also the TTL values assigned to each request. We have chosen these three algorithms to take into account different hypothetical scenarios. The Greedy algorithm provides us with an upper bound for the potentially cacheable content, and the TTL-Cache algorithm the lower bound if all the optional parameters are followed explicitly. However, in reality, caches op-

erate somewhere in between. For instance, some caches serve objects after they become stale.

Figure 9 shows the caches byte hit ratio as we process our data set over time. We found with our Greedy algorithm that 92.2% of requests and 67.9% of bytes could have been served from cache. With the No-Cache algorithm we found 70.1% of requests and 37.4% of bytes could have been served from cache. Finally, with the TTL-Cache algorithm 62.0% of the requests and 32.5% of the bytes would be served from cache. Notice that when taking into account the TTL of the objects that there is only a small 5% change in the bytes served from cache.

We also found that the total cache size using the Greedy and No-Cache algorithms was 17.4TB and 11.3TB, respectively. These cache sizes may be economically infeasible to deploy in all scenarios. However, in today's environment where disk space is relatively cheap and large ISPs have regional data centers (POPs) that serve upwards of 150,000 to 1,000,000 subscribers, the deployment of a 10TB cache is both economically and technically feasible.

Our results allow us to estimate that 30% of Web and Multimedia content is reusable and that 70% of the traffic during the busy hour is web and multimedia traffic, hence about 24% of network traffic could be optimized to take advantage of the fact it is being requested more than once. (We have calcuated the bandwidth savings soley on caching Web and Multimedia traffic, however, a P2P-based cache could also deployed.)

In addition to simulating an unlimited cache size we implemented a caching program to test the TTL-Cache algorithm with various cache sizes. For our simulation, we used Least-Recently-Used (LRU) as our caching policy. The choice of LRU was made because most caching products utilize LRU. In industry this is because it is simple, understandable, and works just as well as any other algorithm when you have enough disk space. Figure 10 shows the results of our simluations using cache sizes varying from 1GB to 1.5 TB in size. The parallel line in the graph shows the maximum amount cacheable bytes (32.5% calcualated previously) if we had an unlimited cache size.

6. NETWORK AWARE FORWARD CACHING

Forward caching has been proposed in the 90s to address the issues of improved client performance and reduced network cost which we highlighted in the previous sections. A forward cache is an HTTP cache deployed within an Internet Service Provider's (ISP) network caching all cacheable HTTP traffic accessed by the customers of the ISP. In contrast to CDNs a forward cache is deployed for the customers benefit and under the control of the ISP, rather than for the benefit of the content owner.

As forward caches are quite frequently collocated with cooperate firewalls caching HTTP traffic of the employees of the cooperation, most large US based ISPs currently do not operate forward caches within there network. The main reason for this decision lies in the economics of deploying forward caches. Forward caches are additional hardware components (typically UNIX servers) which have to be purchased, deployed and managed at a large number of locations. In the US the bandwidth savings can often not justify the cost of such a server deployment.

To reduce the costs associated with forward caching in an ISP we propose *Network-Aware Forward Caching* which is motivated by the insights presented in the previous section. Our goal is to find the set of addresses at each POP that when cached maximizes the cost savings for the network. In particular we noticed that some traffic is already originating close to the ISPs customers (e.g. CDN traffic) and, therefore, the benefits of caching the content again using a forward cache in the ISPs POP is minimal both in terms of performance and cost savings. On the other hand some traffic traverse expensive transit or backbone links and caching would be both cost effective and improve performance. In a second dimension we also noticed that POPs themselves have a high variability in terms of distance to HTTP sources as well as peering links. For example, a remote POP might be far away from a CDN server, whereas a metropolitan POP might be very close to a CDN server.

Considering these insights it becomes clear that the most cost effective way of deploying forward caches is to only deploy them in selected POPs that are caching only selected *expensive-to-deliver* content that is requested frequently. This expensive content can be identified using a metric like air miles as we have done. We call this approach Network-Aware Forward Caching. In the remainder of this section we will formally state the problem of how to place caches and decide what content to cache, propose a solution and evaluate our approach in a case study using data from a large tier one US ISP.

6.1 Problem Formulation

We first define the notations used for the formulation of forward caching problem. The network has a set of POPs $P = \{1, 2, 3, \dots\}$. The distance (air mileage) between POPs are defined as $L = (l_{i,j})$, where $i, j \in P$. The HTTP traffic are downloaded from a set of IP address sets $S = \{0, 1, 2, \dots\}$. For example, an address set can be an address prefix (*i.e.*, 100.200.0.0/24), or the collection of addresses that belong to the same organization or autonomous system (AS). Define $V = (v_{i,j,s})$ as the monthly HTTP traffic volume from address set s that enter the network at ingress POP j and leaves the network at egress POP i. The monthly transit cost per unit volume for address set s is defined $T = (t_s)$, where $t_s > 0$ for provider traffic, $t_s < 0$ for customer traffic, $t_s = 0$ for peer traffic.

Assume we have a total budget of N dollars to purchase and deploy caches. Each cache costs γ dollars, has a disk space of b, and can handle a traffic throughput of e Mbps. We define a boolean variable to denote whether to cache s at POP i: $C = (c_{i,s})$, where $c_{i,s} = 1$ if yes, $c_{i,s} = 0$ if not. We further define $U = (u_{i,j,s})$ as the monthly HTTP traffic volume from s with ingress POP j and egress POP i which cannot be possibly retrieved even from a cache at s with infinite computational power and disk space. We define the disk space needed for caching address set s at POP i as $X = (x_{i,s})$. Note that X is different from U in that an object with size x might have to downloaded twice due to TTL expiration, but just needs x to store.

We define the backbone cost unit as α dollars per mile-byte, and transit cost unit as β dollars per byte. We can then compute the backbone cost, transit cost (TC), and upfront caching cost (CC), when caches are deployed. HTTP traffic $v_{i,j,s}$'s contribution to backbone cost is $\alpha \cdot l_{i,j} \cdot u_{i,j,s}$ when the objects in s are cached at i (*i.e.*,$c_{i,s} = 1$), and $\alpha \cdot l_{i,j} \cdot v_{i,j,s}$ when the objects in s are not cached at s(*i.e.*,$c_{i,s} = 0$). Thus the total backbone cost $BC = \alpha \cdot \sum_{\forall i \in P, j \in P, s \in S} l_{i,j} \cdot ((1 - c_{i,s}) \cdot v_{i,j,s} + c_{i,s} \cdot u_{i,j,s})$. Similarly, transit Cost $TC = \beta \cdot \sum_{\forall i \in P, j \in P, s \in S} t_s \cdot ((1 - c_{i,s}) \cdot v_{i,j,s} + c_{i,s} \cdot u_{i,j,s})$. The total traffic volume at POP i is $\sum_{\forall s \in S, j \in P} c_{i,s} \cdot v_{i,j,s}$, thus the number of cache units at POP i required by the computational power is $\lceil \sum_{\forall s \in S, j \in P} c_{i,s} \cdot v_{i,j,s}/e \rceil$. Similarly, the number of cache units at POP i required by disk space is $\lceil \sum_{\forall s \in S} c_{i,s} \cdot x_{i,s}/b) \rceil$. The upfront caching cost at POP i is the maximum of that required by computational power and that re-

quired by disk space. Thus the total upfront caching cost $CC = \gamma \cdot \sum_{\forall i \in P} \lceil \max(\sum_{\forall s \in S, j \in P} c_{i,s} \cdot v_{i,j,s}/e, \sum_{\forall s \in S} c_{i,s} \cdot x_{i,s}/b) \rceil$.

the problem is to find $c_{i,s}$ such that

minimize: $BC + TC + CC$

subject to: $CC \leq N$

After refactoring, the object function becomes:

minimize: $\sum_{\forall i \in P, j \in P, s \in S} v_{i,j,s} \cdot (\alpha \cdot l_{i,j} + \beta \cdot t_s) - (\sum_{\forall i \in P, s \in S} c_{i,s} \cdot \sum_{j \in P} (v_{i,j,s} - u_{i,j,s}) \cdot (\alpha \cdot l_{i,j} + \beta \cdot t_s) - \gamma \cdot \sum_{\forall i \in P} \lceil \max(\sum_{\forall s \in S, j \in P} c_{i,s} \cdot v_{i,j,s}/e, \sum_{\forall s \in S} c_{i,s} \cdot x_{i,s}/b) \rceil)$

let $B_{i,s} = \sum_{j \in P} (v_{i,j,s} - u_{i,j,s}) \cdot (\alpha \cdot l_{i,j} + \beta \cdot t_s)$, which is the benefit of caching s at i, excluding the upfront cost. The objective function becomes:

maximize: $\sum_{\forall i \in P, s \in S} c_{i,s} \cdot B_{i,s} -$

$$\gamma \cdot \sum_{\forall i \in P} \lceil \max(\sum_{\forall s \in S, j \in P} c_{i,s} \cdot v_{i,j,s}/e, \sum_{\forall s \in S} c_{i,s} \cdot x_{i,s}/b) \rceil \quad (1)$$

subject to :

$\sum_{\forall i \in P} \lceil \max(\sum_{\forall s \in S, j \in P} c_{i,s} \cdot v_{i,j,s}/e, \sum_{\forall s \in S} c_{i,s} \cdot x_{i,s}/b) \rceil \leq \lfloor N/\gamma \rfloor = N'$

6.2 Algorithm

It is easy to see that our final Formulation 1 of the problem for the case that we have only one POP, i.e., $|P| = 1$, is as hard as the *knapsack problem*. In the knapsack problem, given a set of n different items, each with a weight and a value (benefit), our goal is to determine the set of items to include so that the total weight is less than a given limit W and the total value is as large as possible. It is well-known that the knapsack problem is **NP**-hard though it can be solved in pseudo-polynomial time[1] using dynamic programming. In addition, the problem has a polynomial-time $1-\epsilon$-approximation algorithm (an algorithm whose output has a value at least $1-\epsilon$ times the optimum solution) based on dynamic programming, for arbitrary small constant $\epsilon > 0$.

First, let us observe that our problem formulation also has a pseudo-polynomial-time dynamic programming algorithm. Consider any POP i. For a content s, we denote its needed computational power by $C_s = \sum_{j \in P} v_{i,j,s}$ and its needed disk space by $M_s = x_{i,s}$. Now we fill in a table $T[s, C, M]$ which determines the maximum benefit that we can obtain from contents $1, 2, \ldots, s$ with at most C total computational power and M total disk space, where $\lceil C/e \rceil$ ($\lceil M/b \rceil$) is at most the maximum number of cache units that we can afford in our total budget, i.e., N'. $T[0, C, M] = 0$ for all feasible values of C and M. For $s > 0$, $T[s, C, M] = \max\{T[s-1, C, M], T[s-1, C-C_s, M-M_s] + B_{i,s}\}$, for $C \geq C_s$ and $M \geq M_s$, and ∞ otherwise. Next, define $T'^i[U]$ to be $T[|S|, e \cdot U, b \cdot U]$ which is the maximum benefit that we can obtain by caching of contents in POP i with at most $0 \leq U \leq N'$ (as the maximum of computational power or disk space) units of caches. Finally having $T'^i[U]$s, we compute our final table $T''[i, U]$ which is the maximum benefit that we can obtain from POPs $1, 2, \ldots, i$ with at most $0 \leq U \leq N'$ units of caches. $T''[0, U] = 0$ for all affordable values of $0 \leq U \leq N'$, and for $i > 0$, $T''[i, U] = \max_{0 \leq j \leq U}\{T''[i-1, U-j] + T'^i[j]\}$. Therefore, the maximum of our objective in Formulation 1 is $\max_{1 \leq U \leq N'}\{T''[|P|, U] - \gamma U\}$.

By using standard techniques analogous to those for the knapsack problem, it is not hard to transform the above pseudo-polynomial-time dynamic programming to a polynomial-time $1-\epsilon$-approximation algorithm, for arbitrary small constant $\epsilon > 0$.

[1]In computational complexity theory, a numeric algorithm runs in pseudo-polynomial time if its running time is polynomial in the numeric value of the input (which is exponential in the length of the input – its number of digits).

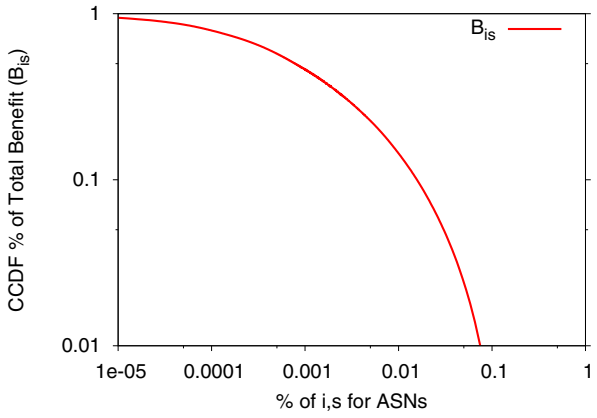

Figure 11: CCDF of Total Benefit at Each i,s Pair

It is worth mentioning that in practice with large values, dynamic programming approaches similar to the aforementioned one in this section are time-consuming and not desirable. Due to this reason we consider a well-known greedy heuristic for the knapsack problem which sorts the items in decreasing order of value per unit of weight and then proceeds to insert them into the knapsack until there is no longer space in the knapsack for more. This heuristic for the knapsack problem is not only very fast and easy to implement but also gives a guaranteed approximation factor 2 for some versions of knapsack. Below we generalize this greedy algorithm for our purpose and report its evaluation results in the next section.

Our greedy heuristics is based on the idea that the total number of caches n is within the range of $[0, N']$. Therefore, we can "guess" and enumerate n. Thus the problem becomes:

maximize: $\sum_{\forall i \in P, s \in S} c_{i,s} \cdot B_{i,s} - \gamma \cdot n$

subject to : $n \leq N'$

By enumerating over all n, $\gamma \cdot n$ is just a fixed cost that can be ignored for the maximization purposes. As we discussed in the previous section, $B_{i,s}$ is the benefit of caching s at i. On the other hand, the weight of content s to be cached on POP i is $w_{i,s} = \gamma \max(\sum_{\forall j \in P} v_{i,j,s}/e, x_{i,s}/b)$.[2]

Now, for a fixed number n of caches, we have essentially a knapsack problem that we want to maximize the benefit of selected elements (i.e., the cache assignment) while our total weight is restricted by the number n of caches. Thus inspired by the aforemention greedy algorithm for the knapsack problem, for a fixed n, our algorithm is to choose the most cost-efficient (i, s) pair (*i.e.*, cache s at i) first. A formal description of our algorithm is as follows:

1: **for** $n = 0$ to N' **do**
2: $c_{i,s} = 0$ for all i and s //clear all $c_{i,s}$
3: **for** (i, s) pairs ranked by $B_{i,s}/w_{i,s}$ descendingly **do**
4: $c_{i,s} = 1$ as long as the total number of used caches so far is not more than n;
5: **end for**
6: **end for**
7: find the lowest $(BC+TC+CC)$ across different n, and output the corresponding $C = (c_{i,s})$;

[2]Note that indeed, $\lceil w_{i,s} \rceil$ is the number of caches needed if we cache content s on POP i alone. This weight might be smaller if we cache other contents on POP i as well. However in some sense we "over-provision" the caches, which is often needed in practice, since we do not want to utilize the caches 100%.

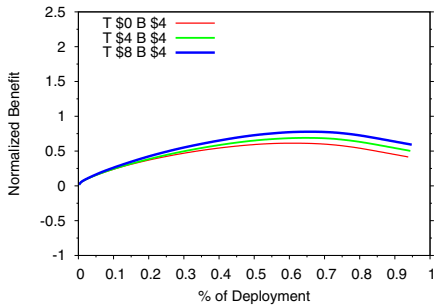

Figure 12: Benefits for Different Transit Costs

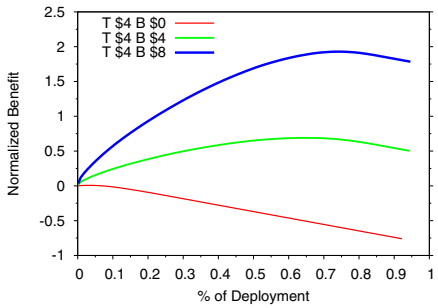

Figure 13: Benefits for Different Backbone Costs

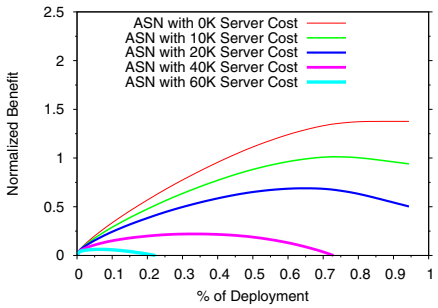

Figure 14: Benefits for Different Server Costs (fixed $4 Transit, $4 Backbone, 400 Mbps, 4TB)

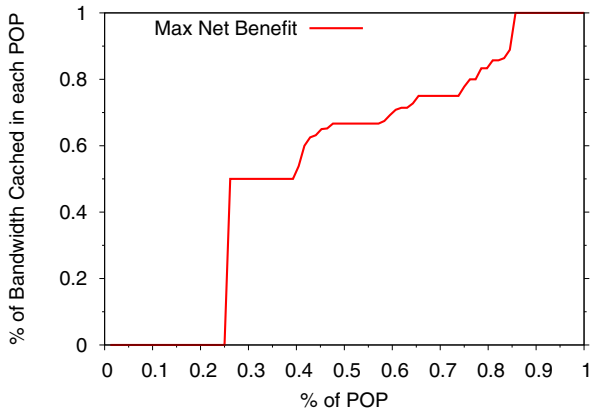

Figure 15: Coverage of Caches at each Pop at Maximum Net Benefit

6.3 Evaluation

We implemented a simulation program to evaluate our network-aware caching approach using the heuristic we described in Section 6.2.

To run the simulation we assume that transit (β) and backbone (α) costs are approximately $4 Mbps/month [30] and the upfront cost of a caching server is $20,000 which can be amortized over 36 month and costs approximately $555 a month to run. We assume that this $20K server will be able to handle 400 Mbps of throughput (e) and as have a disk size of 4 TB (b). We then vary these parameters to study the sensitivity of the results.

In the simulation we base our network specific inputs P, S, $V_{i,j,s}$, and $L_{i,j}$ from our Core Backbone Traffic data set we previously analyzed in Section 4. In our analysis, we have chosen to use AS numbers in S. However, our approach can use other levels of detail such as prefixes if more granular measurements are available. To estimate the values of $U_{i,j,s}$ and $X_{i,s}$ we used our analysis from Section 5 that calculated the values $U_{i,s}$ and $X_{i,s}$ for a single BRAS in this network. In the simulation we use the U and X values from this BRAS as estimates that are scaled appropriately for all other POPs.

Our simulation program calculates the net benefit (Equation 1) as the number of cache servers (n) increases. We find the best deployment solution by selecting n which maximizes the net benefit. In our program, the net benefit is calculated in dollars saved. However, in our figures we plot the relative net benefit. This is done to preserve the anonymity of our data source. Figure 11 shows the CCDF distribution of B_{is}. This shows that most of the benefit can be obtained from caching a small subset of the i, s pairs.

Overall, we find in Figure 12 and Figure 13 that using our stated assumptions for backbone, transit and cache costs and the US Broadband Provider's network data that the maximum benefits would occur when only 68% of the caches are deployed that would have other otherwise been needed to cover all HTTP traffic. Our simulation results show that the relative net positive benefit increased from 0.501 to 0.688 (a 37% overall improvement in benefits) when the network-aware caching approach's strategy was compared to the benefits of deploying caches to cover all POPs. Figure 15 shows the distribution of cache servers to each POP based on our maximal solution. This shows that in the our optimum solution 25% of POPs do not any postive net benefit by having a cache server place there. Only 15% of POPs have a maximumnet net benefit by having all HTTP traffic covered.

Figures 12, 13, and 14 show selected results of varying each of these factors by a couple magnitudes to see their overall affect on the final cost benefit analysis.

Figure 12 shows that transit costs in our simulated network minimally affect the overall maximum net benefit. This is due in our case to the amount of transit traffic that is cacheable is quite low. However, in other network this may not be the case and therefore have a large impact in the results.

Figure 13 shows that the backbone cost is a large factor influencing how much the net benefit is. As the backbone cost increases the cost benefit curve is shifted upwards and maximum net benefit point shifts right.

Figure 14 shows the net benefit cost as the cost of each cache changes but all other parameters remain fixed. When the server cost is set to $0 per server the theoretical maximum benefit obtainable is depicted. As the server cost is increases the net benefit decreases and the optimum number of caches to deploy shifts to the left. This follow intuition that as the cache cost increases that less traffic would have a positive net benefit to cache.

7. RELATED WORK

Forward caches which are also known as proxy caches or forward proxy caches have received a great deal of attention during the dot com boom. As there exists a large volume of prior art on various aspects of forward proxies we refer the interested reader to [24] for a broader discussion of forward caching and focus the related work discussion in this section more narrowly on the problem of cache placement and selective content caching.