

Workload Characterization of Uncacheable Web Content

Zhaoming Zhu, Yonggen Mao, and Weisong Shi

Mobile and Internet System Group
Department of Computer Science
Wayne State University
{zhaoming,ygmao,weisong}@wayne.edu

Abstract

Web caching and content distribution networks (CDNs) are two mainstream approaches to improve Web performance. Unfortunately, we have witnessed the inefficiency of these two approaches because of the rapid growth of uncacheable Web content, resulting from ever-increasing dynamic Web services and cache busting technologies. Although several approaches have been proposed for caching fragment-based dynamic Web content, such as edge side include and client-side include, we believe that a general understanding of uncacheable Web content characteristics is of great importance to future of content caching and delivery.

In this paper, we characterized the uncacheable Web content by collecting an one-day HTTP traffic at a university setting. We found that (1) compared with cacheable content, uncacheable Web content now dominates the HTTP traffic (96.7%), with a larger average size (14K bytes of uncacheable vs. 5K bytes of cacheable); (2) although dynamic content generation consumes longer server processing time, network latency dominates the total user-perceived latency; (3) uncacheable Web content is distributed uniformly among multiple request/reponse pairs in one persistent connection; (4) personalized Web content is less than 10% of total traffic, far lower than previous observations; (5) a considerable portion (42%) of Web content has their TTL values equal to zero. Among this content, we observed URL-alias contributes 38% of total number of requests; (6) the university community has a time-specific access pattern during an one-day interval; and (7) P2P traffic is increasingly scattered among multiple ports (only 13% from default ports for KaZaA traffic), which is a big challenge for P2P caching. Furthermore, based on these observations, we propose several future directions for efficient content caching and delivery of uncacheable content.

1 Introduction

Web caching and content delivery network (CDN) are two major means to improve WWW performance. The efficiency of such performance-enhancing technologies depends on the cacheable property of Web content. Traditional caching mechanism works fine for static cacheable Web objects, but seems futile to this increasingly large portion of today's HTTP traffic: *the uncacheable, probably dynamically generated Web objects*. To face the rapid growth trend of uncacheable content in the HTTP traffic, Web objects with uncacheable property deserve further investigation and exploitation of their cacheable possibility.

Several caching methods have been proposed to deal with these uncacheable Web objects. They could be broadly categorized as *content caching* and *function caching*, which could be used for either server side proxies or client side proxies. In the content caching, the HTML page which the application generates is cached as different fragments/channels. The representatives of this approach include Akamai [1], CONCA [34], and Client Side Include (CSI) [29]. In the function caching, the application itself is replicated and cached along with its associated applications so that the edge servers run applications instead of the origin server. Examples of this method are vMatrix [4], IBM Websphere [19], Active Cache [9].

The successfulness of these techniques depends on the understanding of the uncacheable content, including content characteristics and access patterns. In this paper, we tried to answer these following questions: Among the huge content delivered on the Internet, what portion is cacheable and what portion is uncacheable? What are their characteristics, especially for uncacheable content? Is there any difference among different uncacheable HTTP content? Is there any cacheable possibility for these conventional uncacheable content? How about the cacheability of personalized content? Is there some relationship between uncacheable content and HTTP persistent connection? To our best knowledge, this work is the first effort to characterize the uncacheable Web content.

In this study, we sniff and analyze all incoming and outgoing Internet traffic at Wayne State University (WSU), a medium-sized educational institution with 35,000 students, faculty and staff. The *tcpdump* [37] is used on the campus gateway switch to sniff TCP packet. For the traffic reconstruction purpose, we build WebTACT, an offline Web traffic analyzer application, to reconstruct the TCP streams and the corresponding HTTP sessions.

In this paper, we traced one day traffic and see over seven million request/response pairs and 66 gigabytes HTTP data. Our analysis shows that: (1) uncacheable data have dominated the majority portion of today's HTTP traffic (96.7% of total transferred Web content), and multimedia type content transferred by P2P applications (44.3%) and graphic type (jpeg and gif) content (18.2%) are top two in the uncacheable Web objects; (2) compared to cacheable content, uncacheable content takes more server-processing time. But due to network latency, the client-perceived response time tends to be close to that of cacheable content; (3) on average, uncacheable content have an larger object size than that of cacheable objects (14 K bytes vs. 5K bytes); (4) clients that access personalized Web content and servers that provide personalized content are more concentrated than general clients and server groups, while the total online personalized content occupies only a smaller percentage (less than 10%), far below than previous observations from [13, 41]; (5) the university community has a time-specific access pattern during an one-day interval; (6) uncacheable Web content are distributed uniformly among multiple request/reponse pairs in one persistent connection; (7) a considerable portion (42%) of Web content has their TTL values equal to zero. Among these content, we observed URL-alias contributes 38% of total requests; (8) P2P traffic is increasingly scattered among multiple ports (only 13% from default ports for KaZaA [21] traffic), which is a big challenge for P2P traffic caching.

Based on these observations, several implications could be drawn: (1) a considerable portion of uncacheable Web content is cacheable; (2) domination of network latency factor motivates the moving of functionality for uncacheable Web content generation to the edge of network; (3) observed Web access pattern makes time-specific prefetching a promising solution; (4) prefetching for personalized Web content is promising because of the popularity of clients and servers; (5) convinced by the observed P2P request popularity, we believe that content-based caching is significant to the ever-increasing P2P traffic; (6) exploiting URL-alias is a promising direction to improve cacheability of uncacheable content.

The rest part of this paper is organized as follows. The next Section gives out the background of web content classification. Followed is the section that describes the methodology used in our study. Section 4 starts with a high level characteristics analysis, followed by a detailed discussion of the cacheable and uncacheable content traffic patterns and meaningful features. Section 5 discusses the possible implications to Web content caching and delivery. Related work and discussion are described in Section 6. Finally Section 7 lists the summary and future directions.

2 Background

From the viewpoint of Web caching, generally Web content could be broadly categorized as uncacheable Web object or cacheable Web object, as shown in Figure 1 (Web object and Web content all refer to the "real stuff" transferred via HTTP protocol and will be used interchangeably in this paper). The cacheable Web content refers to those infrequently changed Web objects (also known as static web content).

The uncacheable Web content could be further subcategorized into 7 uncacheable subtypes, depends on the following rules:

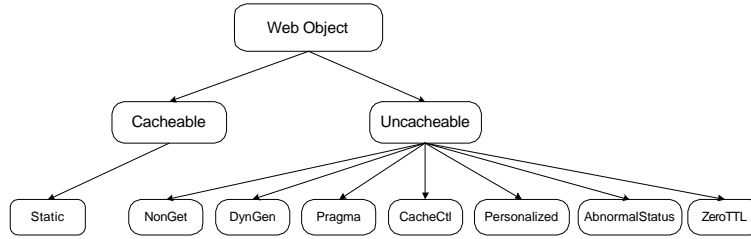


Figure 1: Web object categorization.

Subtype 1 - NonGet: If the HTTP method, appeared in the HTTP request header, is not a GET method, then the Web object would be classified as NonGet subtype;

Subtype 2 - DynGen: If the method is “GET”, but the request URL contains keywords (like “cgi”, “asp”, “=”, “?”, ...etc), which implies a dynamically generated Web object, then the Web object would be classified as DynGen subtype;

Subtype 3 - Pragma: In the cases that HTTP request/response header part contains “Pragma: no cache” control information header, this Web object could be regarded of Pragma subtype;

Subtype 4 - CacheCtl: In the case that HTTP request/response “Cache Control” header contains information indicating this is a dynamic, uncacheable Web object, this Web object is classified as CacheCtl subtype;

Subtype 5 - Personalized: If the HTTP request header contains Cookie header, this object is defined as a personalized content;

Subtype 6 - AbnormalStatus: If the return status code, from server, is not 2XX, we think the response object is not a normal response and treat it as of AbnormalStatus subtype;

Subtype 7 - ZeroTTL: If the Web object does not belong to the above six subtypes, but the calculated TTL(Time To Live) value of the response object is zero, this kind of object is classified as ZeroTTL subtype (Detail TTL calculation algorithm are given out in methodology section).

3 Methodology

In our study, *tcpdump* is used to collect TCP packets at the network entrance of WSU. To capture all the HTTP traffic, all TCP packets are sniffed, except some well-known service ports (eg, ports for FTP, Telnet, SSH, ...etc). Analysis results in Section 4.2.8 verify that P2P traffic, based on HTTP protocol, is not limited to their default application ports (for KaZaA application [21], only 13% traffic is through its default port 1214).

To extract the complete HTTP information, including both header and content, we have developed WebTACT, an **Web Traffic Analysis and Characterize Tool**. WebTACT consists of three parts, as shown in Figure 2. In the *TCP session reconstruction* part, TCP sessions are reconstructed first. Our algorithm is very similar to that proposed in [12]. One TCP session is identified by the first several TCP packets(SYN, SYN/ACK, ACK), used in 3-way hand-shaking period. Without these initial TCP packets appeared in dumped data, we can not find the start of a TCP session and rebuild it. With these initial TCP packets, we fix the source IP address and port, destination IP address and port and the beginning sequence number and acknowledgement number for this specified TCP session. Followed TCP packets are determined to belong to this session, based on their source/destination IP address and port numbers combination, and whether their captured time lies within the 2MSL(Maximum Segment Lifetime, we choose 60 seconds in our analysis) time span with the latest TCP packet in this session, are gathered into this

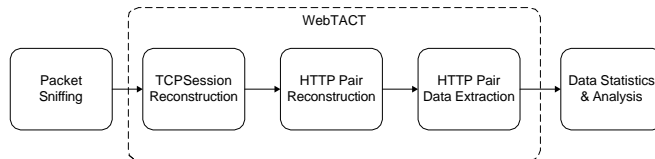


Figure 2: A diagram of data collection and reconstruction.

specific TCP session. After the TCP session is rebuilt, All the TCP packets payload are accumulated to rebuild the buffer content, in the *HTTP pair reconstruction* part. Then in the *HTTP pair data extraction* part, we extract out the interesting information from the corresponding HTTP headers, calculate the hash digest values for the `Cookie` field and the Web content, for our analysis purpose. Our WebTACT analyzer is written in C++, running on Linux RedHat, kernel version 2.4.18. The sensitive information, such as IP address, is anonymized and stored on isolated storage device.

The concept of “pair” is used in our characteristics analysis. A pair is an request/response pair consists of one original HTTP request and one corresponding HTTP response. For the HTTP version 1.0 and 1.1, there are features of keep-alive connection and pipelining. This means, the payload of one TCP session could contain multiple requests/response pairs. In this situation, based on the assumption that the order of multiple requests/response are matched perfectly [25], we regard each corresponding HTTP request/response as an HTTP request/response pair and the whole TCP connection consists of a list of this HTTP request/response pair).

TTL (time-to-live) is defined as the difference between web object freshness lifetime and its age. The age of an object is the difference between current time and the time specified by the `Date` header field. In our calculation, the age is always zero except when it is specified by the `Age` header. So the TTL of a Web object is equal to its freshness lifetime if its age is zero. We use Squid’s [36] implementation to calculate freshness lifetime. First, if a `max-age` directive is present, the value is used as the freshness lifetime, and the TTL is the freshness time minus the age if it is present by `Age` header (or zero if negative). Second, if `Expires` header is found, the freshness time is the `Expires` time minus the `Date` header field time (zero if negative). Otherwise, the origin server provides no explicit freshness lifetime and a heuristic is used: The freshness lifetime is a fraction (20%) of the time difference between the time specified by its `Date` header and the time specified by its `Last-Modified` header.

4 Analysis Results

We collected one-day period (12:00 pm, Mar 18 —12:00pm Mar 19, 2003) Web traffic,¹ rebuild and investigate the contained HTTP traffic. The analysis results and observations based on our experimental data are give in this section.

4.1 High Level Characteristics

Table 1 lists the high level statistics for both cacheable and uncacheable content. For each content type, we detail them in different traffic directions. The inbound traffic means the response Web objects are targeted to clients inside WSU campus, while the outbound traffic means that the response Web objects are targeted to clients outside WSU campus. The total distinct client number inside WSU campus is 8,889, and that outside WSU is 85,710. The total server number inside WSU is 1,846, and that outside WSU is 105,932. The ratio of inbound traffic vs. outbound traffic is about 4 : 3, while the corresponding ratio in University of Washington’s trace is 1 : 5 [31].

¹Due to the diary-based access pattern observed in previous analysis ([17], [32]), we are convinced that one-day trace is enough for this analysis.

Type	Cacheable		Uncacheable	
	Inbound	Outbound	Inbound	Outbound
HTTP Traffic Direction				
# of Servers	4963	75	105722	1846
# of Clients	6625	6251	8880	85425
Total Gross Traffic(bytes)	1,935,171,426	350,864,476	38,304,575,996	30,061,284,040
Total Object Size (bytes)	1,664,000,967	319,712,501	33,567,629,788	28,156,567,593
# of Requests	314,480	42,553	5,399,513	1,991,338

Table 1: High-level statistics of HTTP traffic.

From Table 1, we can see that, the captured-reconstructed gross HTTP traffic (include HTTP headers and bodies) is around 65.8 G bytes. For the total objects size (or the total transferred HTTP response objects’ sum bytes; all the traffic mentioned later in this paper are referring to total objects size), the uncacheable content outnumbers the cacheable content (57.5G Bytes vs. 1.85 G Bytes). The servers providing uncacheable content outnumber those providing cacheable content (107,568 vs. 5,038) , while the clients accessing uncacheable content also largely outnumber the clients accessing cacheable content(94,305 vs. 12,876). These data exemplify that the uncacheable content is today’s dominating HTTP traffic portion. Figure 3 gives out the top 15 cacheable/uncacheable content types, in total bytes and request/response numbers, and in both traffic directions. As shown in Figure 3(a), the majority of HTTP uncacheable traffic is multimedia audio/video type. This is because that: (1) the huge volume of P2P KaZaA application traffic focuses mainly on multimedia file exchange; (2) request URLs of KaZaA contain “=”, so it is categorized into DynGen (subtype 2) uncacheable content type. For P2P HTTP traffic, a more and more “hot” Web application topic these days, we have our detailed analysis in Section 4.2.8.

The reason for the large percentage of “Unknown content type” in Figure 3(b) is that a large number of responses (2,700,803, 36.5% of total 7,390,851 uncacheable responses) are with zero Web object size, so that we have to classify their type as “Unknown content type”. These zero-size object cases include large portion of responses with abnormal return status code, that of `AbnormalStatus` subtype.

Comparing our data with previous results in [41], we observe that an increase in the uncacheable request/response for image (`gif` and `jpeg`) content type, and a decrease in text (`html` and `plain`) content type. The multimedia type Web objects (`video/x-msvideo`, `video/mpeg` and `audio/mpeg`), with a large traffic percentage and a small request/response percentage, implies a large average size of these kinds of objects.

4.2 Detailed Characteristics of Uncacheable Web Content

After describing the breakdown of different uncacheable subtypes first in this section, we in turn present the response time and object size distribution, popularity and access pattern, P2P traffic, and other interesting features of uncacheable content as follows.

4.2.1 Uncacheable Content Breakdown

Figure 4 shows the percentage breakdown for each of the seven uncacheable subtypes, by their total object size and request/response number, and Table 2 lists the detail absolute numbers. The “combined type” means the uncacheable subtype is a combination of this subtype and at least one other uncacheable subtype, while the “pure type” means the request belongs to this subtype only, not mixed with other subtypes. From Figure 4, we found that the personalized Web objects (subtype 5) consists of less than ten percent of all of the uncacheable Web object traffic and requests, not as large as previous observation [12, 42]. We do not know the reason for this low percentage (both total object traffic and request number) of personalized Web objects.

Among the DynGen traffic, P2P (KaZaA) traffic occupies a considerable portion (76.2%). Another large portion, `ZeroTTL` (subtype 7), implies a promising probability of caching performance improvement that we will give more detail analysis later.

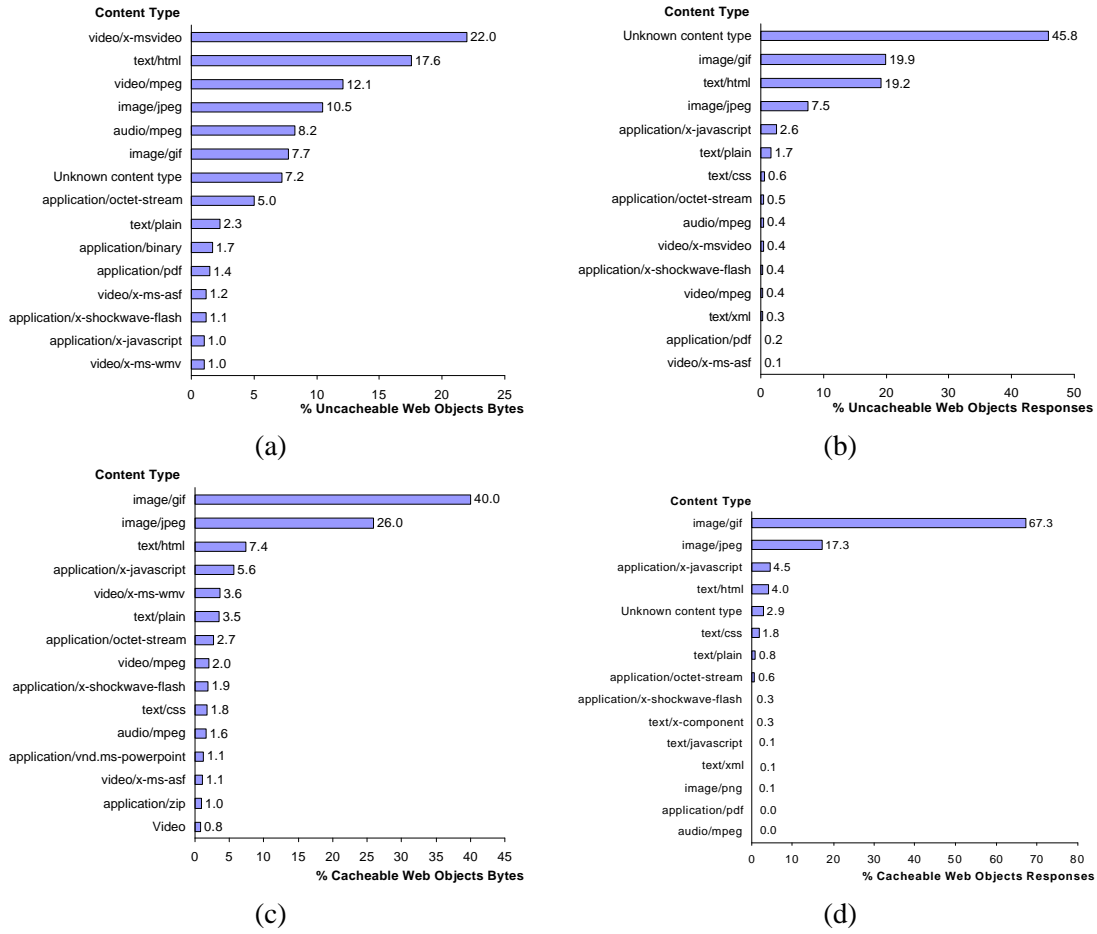


Figure 3: Histogram of top cacheable/uncacheable Web content type by traffic bytes and responses.

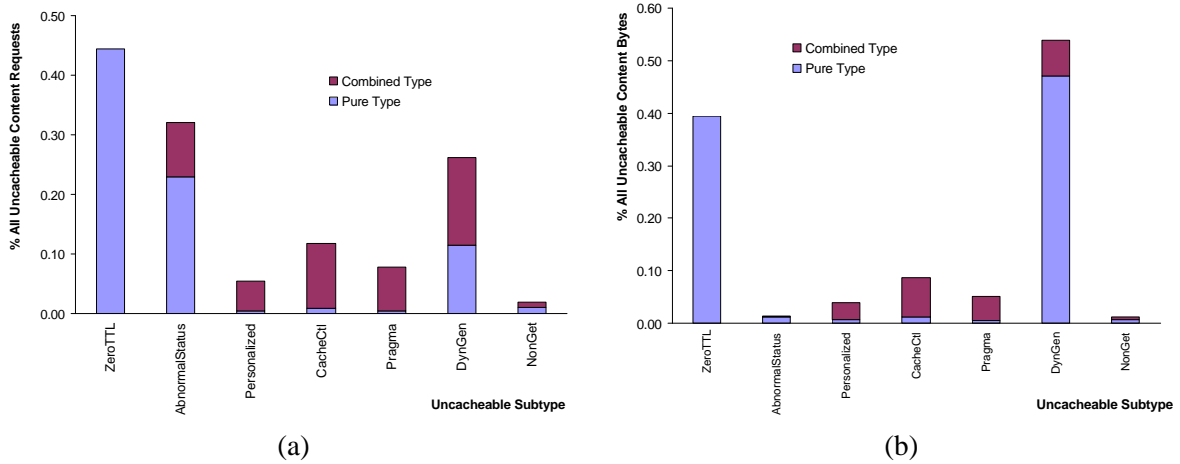


Figure 4: Histogram of all 7 uncacheable Web objects subtypes by object size and requests.

Subtype	ZeroTTL	AbnormalStatus	Personalized	CacheCtl	Pragma	DynGen	NonGet
pure requests	3,277,657	1,690,412	30,797	65,693	30,905	848,420	71,159
pure bytes	24,295,285,805	683,242,184	456,549,536	779,787,204	283,069,280	29,090,380,992	386,743,519
combined requests	0	675,815	373,062	807,348	550,062	1,086,120	71,248
combined bytes	0	200,424,533	2,004,427,341	4,587,394,559	2,886,758,135	4,179,517,396	329,551,841

Table 2: Detail breakdown for all 7 uncacheable Web object subtypes.

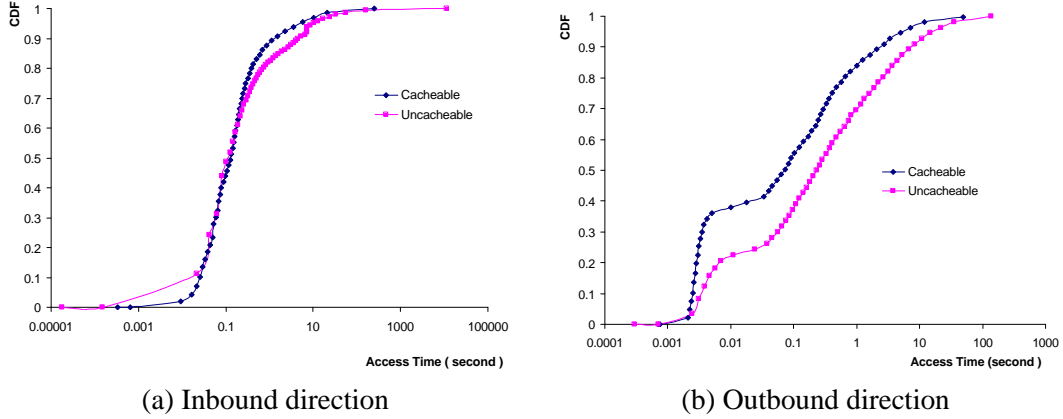


Figure 5: CDF of response time for cacheable/uncacheable Web objects with different traffic direction.

4.2.2 Response Time and Breakdown for Uncacheable Content

For further analysis, we first want to know, whether the cacheability of Web objects affects their response time, on both server side (processing time) and client side (latency). In our study, The time difference between the TCP packet containing the first byte of HTTP request and the TCP packet containing the last byte of the corresponding HTTP response, is calculated as the response time. The timestamps on these TCP packets were tagged when these packets are collected. *Tcpdump* could record the timestamp with microsecond level precision, which is much more accurate than that obtained from conventional server or proxy logs.

Figure 5 shows the cumulative distribution function (CDF) of response time for cacheable/uncacheable Web objects, in both inbound and outbound directions. In this figure, the x-axis represents the response time in ascending order, and the y-axis represents the cumulative percentage of responses for the corresponding x-axis value. Because of the sniffing position of our study, we could assume that for the inbound traffic, the response time is close to client-perceived latency, and for the outbound traffic, the response time is close to server-processing time.

For the outbound HTTP traffic, there is a clear difference between the response time of uncacheable and cacheable content, this is probably caused by the time necessary to dynamically generate the uncacheable content.

For the inbound HTTP traffic, the difference between the response time of uncacheable and cacheable Web objects is not so clear. This implies that the time difference caused by dynamic/static content generation has been blurred by the network latency on their route.

Figure 6 shows the CDFs of response time for the six subtypes uncacheable Web objects. Pure subtype data sources are applied to avoid interference from other subtypes. To isolate the effect of large size KaZaA objects, we separate the KaZaA application data from DynGen (subtype 2) data, so that DynGen curve in the figure represents the real dynamically generated content, while KaZaA curve represents the KaZaA objects' access time. In Figure 6(b), the data sample for **Pragma** subtype 3 is too small (only 11 samples exist), so the curve for subtype 3 is not very meaningful. Figure 6 show that except large size objects (KaZaA) and **AbnormalStatus** (subtype 6), access time of other subtypes of uncacheable objects are quite close. **AbnormalStatus** object implies the servers response with some status code reflecting the abnormal return status, which obviously costs the least computing and takes the least time.

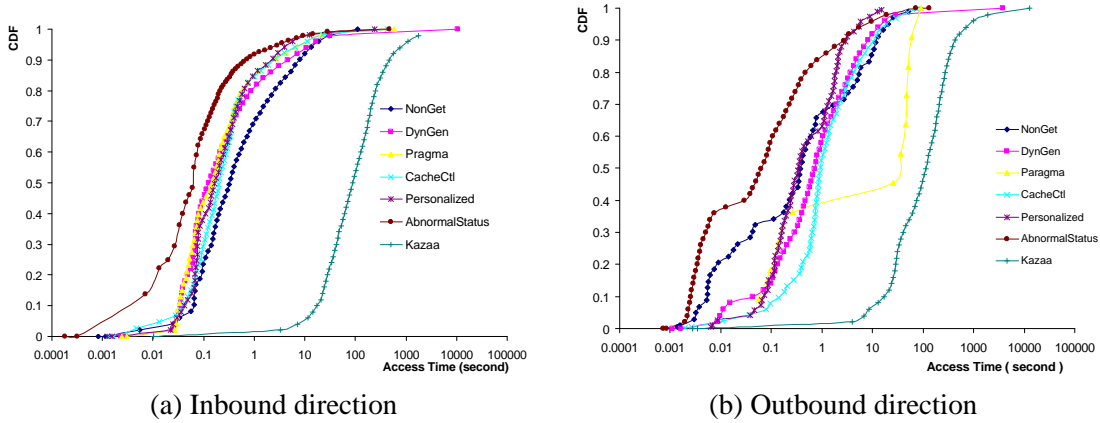


Figure 6: CDF of response time for six uncacheable subtypes Web objects (with KaZaA separated) in both directions.

4.2.3 Object Size Distribution

Object size distribution is also an interesting topic of our study, especially when Web objects are classified into two major classes: cacheable and uncacheable. Figure 7 shows the CDF for Web objects size. For uncacheable objects, a large portion (40%) of object size is zero, so we exclude these zero-size objects in the data source of Figure 7. Intuitively, we believe that, on average, uncacheable object size is smaller than cacheable size, but our analysis gives contrary result. For cacheable and uncacheable objects, Figure 7 shows that 90% of cacheable objects smaller than 11,742 bytes, while same percentage uncacheable objects are smaller than 19,167 bytes. The average size is 5K bytes (cacheable) vs. 14K bytes (uncacheable).

Amazingly, the largest Web object size we observed is 83M bytes for uncacheable object and 30M bytes for cacheable objects. These numbers are much smaller than that appear in [31]. The possible reasons are: (1) our data collecting period is relatively short (24 hours vs. 9 days data collecting period of [31]); (2) our WebTACT analyzer calculates the Web object size based on the real reconstructed Web object, and for real large Web objects, like the multimedia video objects with size over 100M bytes order, HTTP 1.1 application supports the transferring of partial objects (most P2P applications do transfer fragments of these large multimedia objects, via multiple processes or threads, to achieve satisfied application performance). So the real reconstructed (fragmented) objects reflecting a fraction of total size is a reasonable explanation.

4.2.4 Is Uncacheable Content Really Uncacheable?

Although the object composition technique, such as ESI [38], CSI [29], has been proposed, in this paper we are looking for URL-alias derived cacheable possibility. There exist a large number of Web objects that do not belong to the six uncacheable subtypes, but their calculated TTL value is zero, so we classify them as ZeroTTL subtype.

Figure 8 shows the CDF of ZeroTTL objects digest, based on their digest value repeatness rank, not on their URLs. There are total 3,277,657 Web objects belonging to ZeroTTL subtype. Among these, almost 27% (896,086) are zero-size objects, that are excluded in the data source of Figure 8. In Figure 8, we observe many that different URLs share the identical content digest. This is caused by the phenomenon called “URL-alias” [23]. The figure shows that the 1st rank repeated digest value repeats 18,746 times, the 100th repeats 1,038 times, and the 1000th repeats 113 times. The number of requests targeting the top 1000 (less than 0.1% of total distinct digest value) rank digest value count for 22% of total number of requests. This observation reveals an opportunity for the future Web cache improvement if certain protocol could be designed to deal with ZeroTTL objects based on their digest value, rather than on their URLs only.

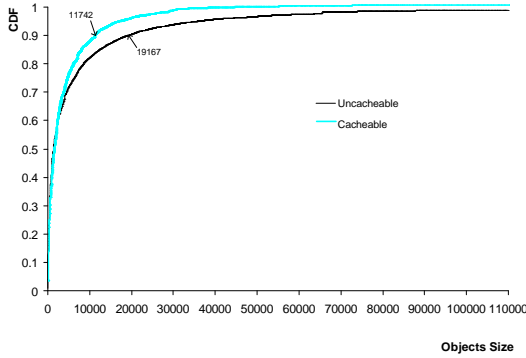


Figure 7: Size distribution for cacheable/uncacheable Web objects.

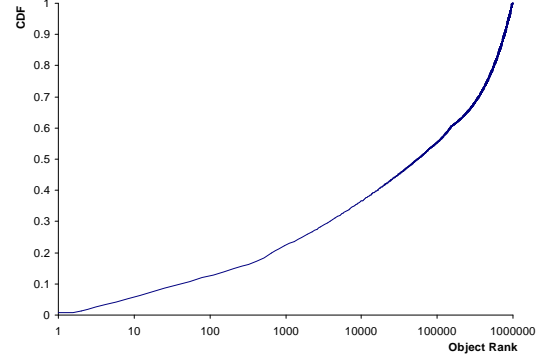


Figure 8: CDF of repeated digest for ZeroTTL Web objects.

4.2.5 Popularity Analysis

Client/Server Popularity Figure 9 and 10 plot the client/server popularity, when accessing/providing personalized content and general web content. We assume that the personalized Web content would be more client/server-specific, than general Web content, due to its “personalized” property. And these figures do verify our assumption.

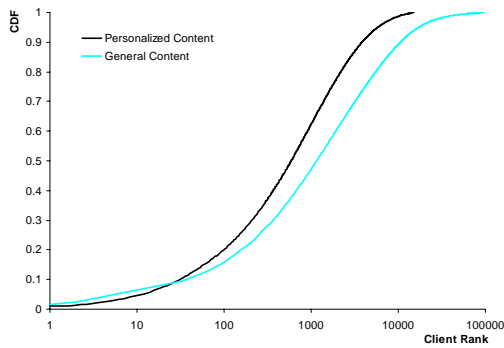
Figure 9(a) shows that clients consuming personalized content are more concentrated than the general clients population. Top 1% of clients accessing personalized content bring about 20% of the total personalized content requests. However, unlike previous observations [13, 41], we find that clients interested in personalized content only occupy 10% of the total client population. In Figure 9(b), we plot the requests distribution of the top 100 clients that access outside personalized Web content. The graph reveals that some clients are much more likely to access personalized Web content. These clients are some public-access computers, located at public area like student dormitories, for students check updated personalized information like email or personal account on e-commerce Web sites.

Figure 10(a) also shows that personalized content is provided by 10% of the total servers, and servers providing personalized content are also more concentrated than server providing general content. Top 1% of servers that provide personalized content handle 69% personalized content requests. In Figure 10(b) we plot the request distribution of the top 100 outside servers. The graph shows the existence of the “hot” personalized Web servers and the top 30 of the servers contribute 71% of the total requests among the top 100 servers providing personalized content.

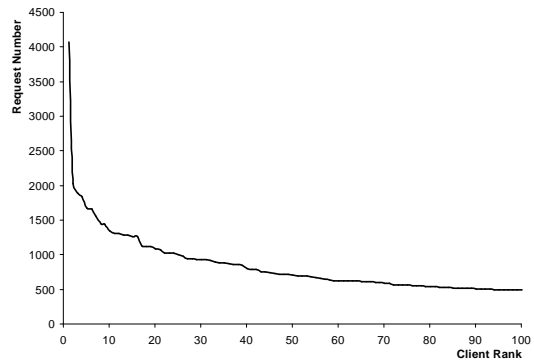
Object Popularity Due to the personalized property, personalized Web objects would not be more concentrated than general Web objects. Figure 11 plots the object popularity for general content and personalized content, and verifies our assumption. It also shows that personalized content only occupy less 10% of the total Web content, as already shown in Figure 4 and Table 2.

4.2.6 Persistent Connection vs. Uncacheable Web Objects

In our collected-reconstructed HTTP data, there are totally 551,952 persistent connection sessions, consisting of 3,346,127 HTTP request/response pairs. On average, a persistent session consists of 6.06 pairs. The longest TCP/HTTP connection session consists of 1085 HTTP request/response pairs. We have supposed that the uncacheable content would have some distribution patterns among multiple pairs within one persistent connection. One reasonable assumption is that, for a persistent HTTP connection, maybe the first object is an uncacheable dynamically generated page template, followed with embedded cacheable objects like graphics [6]. But Figure 12 shows that, in the first 10 pairs of persistent connections, the ratio of uncacheable objects number to the total object number is around 93-95%; and the ratio of total captured uncacheable objects number to total captured objects number is 95%. So we believe that there is no special distribution pattern for uncacheable content among pairs in

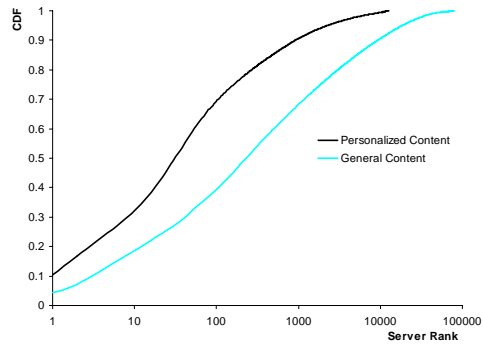


(a) Client popularity

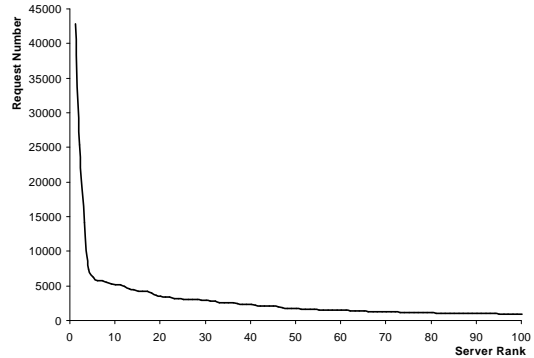


(b) Top 100 rank inbound clients distribution

Figure 9: Clients popularity for general content and personalized content.



(a) Server popularity



(b) Top 100 rank inbound servers distribution

Figure 10: Servers popularity for general content and personalized content.

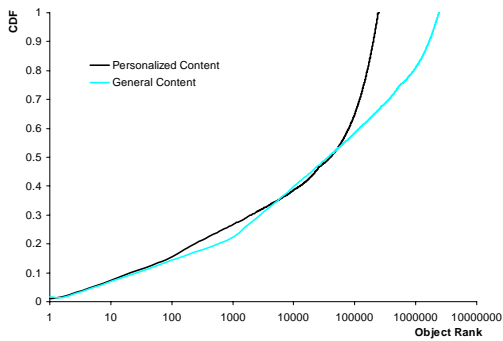


Figure 11: Web objects popularity.

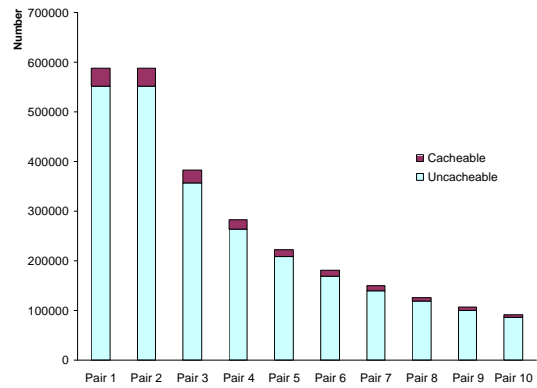


Figure 12: Uncacheable/cacheable distribution in persistent connection.

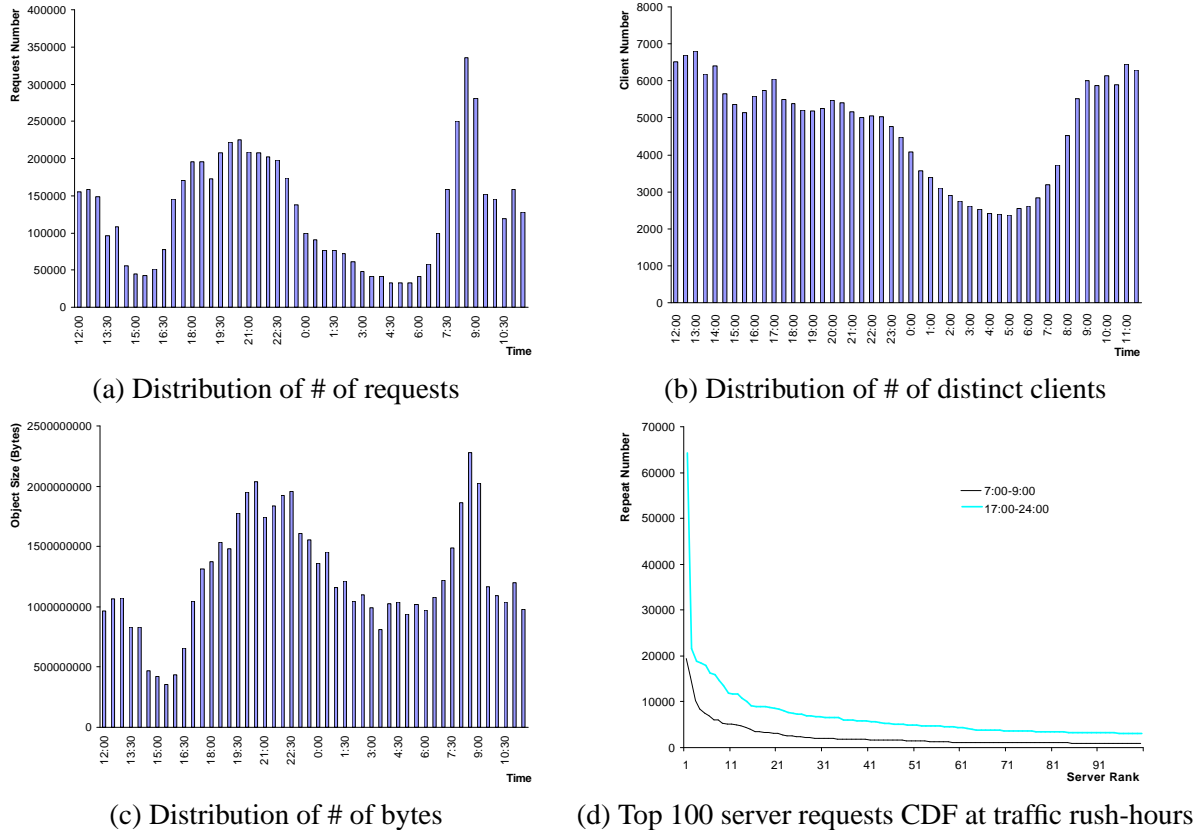


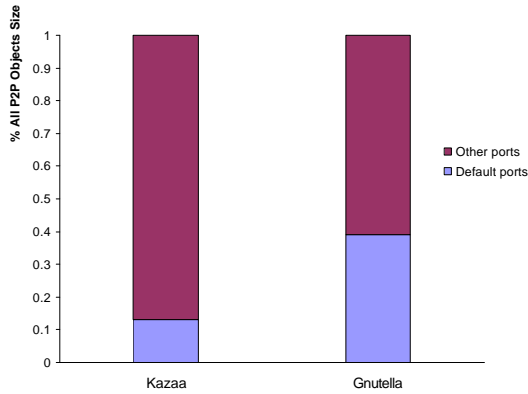
Figure 13: Access Pattern during 1-day period.

a persistent connection. In other words, uncacheable content is distributed uniformly along the multiple pairs of a persistent connection. One possible reason is cache busting technology [25].

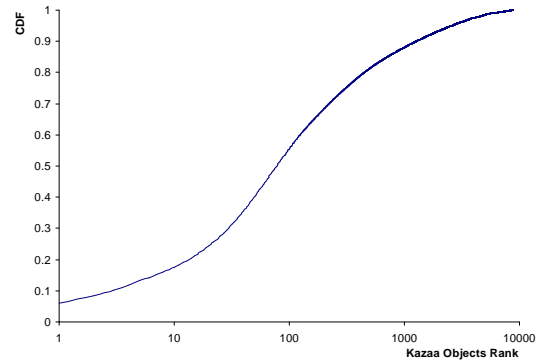
4.2.7 Access Patterns of Uncacheable Web Content

Clients Web accessing pattern is also an interesting issue that we want to look at. Figure 13 shows the total HTTP request numbers, total distinct client number online, and total Web object traffic volume, by every half hour time interval. We could see that for WSU Web traffic, there are clear peak period around 7:00am-9:00am and 17:00pm-24:00pm. time slots. For the 7:00am-9:00am period, the traffic get to a peak value, shows the clients' morning Web-browsing habit. For the 17:00pm-24:00pm period, the relative small client number and request number, with relative large traffic size, reveal a relative heavy traffic per client/per request during that time period.

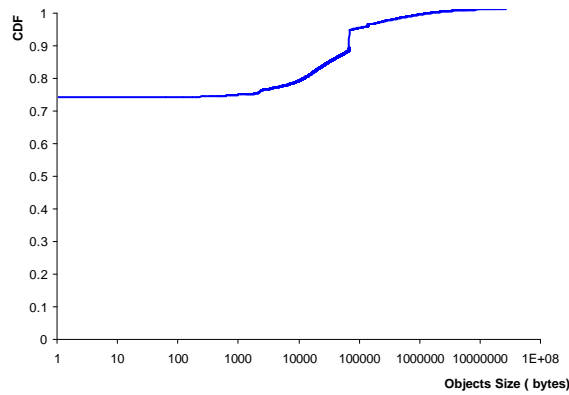
Based on this obvious time-specific access pattern, we investigate the requests handled by top 100 outside servers during these two rush-hour time slots. Figure 13(d) gives out the CDF of the number of requests targeting for top 100 servers, during the above mentioned two time slots. It displays the high concentration trend for these outside servers. In the morning time slot, top 100 popular outside servers contribute the 38.6% of total HTTP request for outside servers, and top 100 popular outside servers in the evening time slot contribute 30%. Note that though there are a sizeable concentrated servers in these two time slots, only 55% of them are overlapped. This implies a possible prefetching policy based on the time-specific access pattern and personalized behaviors.



(a) Web objects size by P2P applications using different ports



(b) KaZaA objects popularity



(c) CDF of KaZaA object size distribution

Figure 14: P2P applications.

4.2.8 Peer-to-Peer Traffic Analysis

With WebTACT, P2P traffic is captured and reconstructed on all possible ports (except some well-known ports), instead of on some specific default ports only (1214 for KaZaA , 6346 and 6347 for Gnutella, used by previous work [31]).

For Gnutella type P2P applications [16], we find several different Gnutella client applications (LimeWire, BearShare, Shareaza, Gnucleus, Gnutella, gtk-gnutella, Morpheus, Mutella, MyNapster, XoloX) appeared in HTTP traffic and aggregate their traffic into a whole Gnutella division. For KaZaA type data, we calculate the traffic from only one client program: KaZaA client. The total Web object size transferred by Gnutella applications is 1,087,336,501 bytes, while that by KaZaA application is 25,363,680,167 bytes. The total object size transferred by P2P applications occupies 41.5% of the total observed Web object size while the corresponding percentage is over 75% in [31]’s work.

Excluding zero-size object, the average object size of P2P objects is 260K bytes, while that of WWW (without P2P) objects is 7.7K bytes. On average, P2P object size is two order of magnitude larger than that of WWW (without P2P) objects, while [31] indicated that P2P object size is three order of magnitude larger than that of WWW (without P2P) object size. We ascribe this to the partial content delivery mechanism provided by HTTP 1.1 protocol [14].

Figure 14(a) shows that, with the evolution of P2P applications, P2P traffic ports are more distributed than before. For example, only 13% of KaZaA traffic is through its default port 1214. This port scattering phenomenon would bring a challenging for efficient P2P traffic caching.

Figure 14(b) shows the object popularity in KaZaA traffic. We can see that , KaZaA objects are highly concentrated, which is the same as the observation in [31]. Requests for top 100 (out of 8673) most popular KaZaA objects account for 55% of the total object requests. This phenomenon shows: (1) there are probably “hot spots” in P2P traffic; (2) the P2P applications probably apply multi-process/multi-thread communication to improve application performance, thus lead to multiple requests to the same Web object, for their different fragments.

Figure 14(c) plots the CDF of P2P object size distribution. The CDF curve consists of two parts. The first is 73% zero-size objects, that implies an abnormal return status, or P2P protocol related content. This observation is similar to that in [31]. The second part shows nearly 27% objects with size larger than 1000 bytes. In this part, there is a very concentrated part (vertical line in graph, occupying 5.7% of total objects), which corresponds a size range from 64 K bytes to 69 K bytes. This phenomenon strongly implies a prevailing partial object transmission happened in P2P application traffic.

5 Implications for Caching and Delivery of Uncacheable Content

The analysis of WSU HTTP traffic indicates both the need to improve delivery of uncacheable web content, and the opportunity to cache the uncacheable web content and P2P traffic. We discuss these implications below:

- **Need for efficient delivery of uncacheable content** The growing popularity of P2P applications and various dynamic and personalized content have resulted in uncacheable web content becoming an important part of current-day HTTP traffic. Our study has shown that 96.8% of HTTP traffic is uncacheable web content. Unfortunately, traditional web caching and CDNs developed to improve delivery static content do not yield the same benefits for uncacheable content. This situation will lead client-perceived latencies to increase and loss of network bandwidth among various network applications.
- **Is uncacheable really uncacheable?** Although we observed most of HTTP traffic was uncacheable Web content, but 39.4% of the uncacheable content whose subtype is ZeroTTL has the repeatness based on their hash-based digest values, resulting from the “URL-alias” phenomenon. The repeatness provides an opportunity for caching if certain protocol could be designed based on the digest value. In addition to the object composition technique, such as ESI [38], CSI [29], we believe a considerable part of uncacheable content is cacheable.
- **Need for migrating the dynamic content generation functions to the network edge** Our results show that client-side perceived response time of uncacheable content is very close to that of cacheable content, while the uncacheable content will need more server processing time to generate than cacheable content. The possible reason is that the network latency blurred the difference. This implies that the further server-side effects will not be perceived by the client-side, and one possible solution is migrating those dynamic functions to the network edge. Our initial work on generating personalized emails at the network edges shows a significant performance improvement [30].
- **Prefetching for personalized content** Clients that access personalized content are more concentrated than general clients, like at library, student center and dormitories, and servers that provide personalized-content also show same concentration. Those show that personalized prefetching could be used at those organizations to reduce the client perceived latency and network bandwidth equipment.
- **Access patterns based prefetching** WSU Web traffic has some access patterns. In Figure 10, we found there were peak period around 7:00am-9:00am and 17:00pm-24:00pm time slots. The different time slot had different web-browsing focuses. This implies that a perfecting based on different time slot access pattern could be implemented to reduce popular servers load and user-perceived latencies. In addition to a diary-based access pattern, exploiting the geographically distribution of the popular machines to deploy prefetching services is an interesting direction to improve prefetching hit ratio in general.

- **Potential for P2P traffic caching** The P2P traffic accounts for 41.5% objects bytes of total TCP traffic, and our analysis shows that KaZaA objects requests are highly concentrated. The top 100 objects account for 55% objects requests. Most of them are fragments and belong to the second subtype DynGen of uncacheable content. As such, they could be cached by the content-based caching. This will significantly reduce the bandwidth consumption of P2P applications, and reduce the local traffic within the organization. However P2P traffic is increasingly scattered among multiple ports (only 13% from default ports for KaZaA traffic), which is a big challenge for P2P traffic caching.

6 Related Work

Web workload characterization has been extensively studied in the past from the perspective of proxies[7, 11, 40, 42], client browsers[5, 10, 22], and servers[3, 27, 35]. However, many of these studies focus on the characteristics of static (cacheable) Web content, while this work focuses on the characterization of uncacheable Web content. To the best of our knowledge, the work presented in this paper is one of the first efforts that attempting to understand the access patterns to uncacheable Web content in a general context. The work presented in this paper compliments previous work on understanding the characteristics of dynamic and personalized Web content [33, 35].

The methodology used in our analysis is very close to that of previous work [13, 15, 31, 41, 42], however, the analysis emphasis on uncacheable Web content in this paper distinguishes our work from these previous work. For example, Wolman et al. analyzed the Web traces of University of Washington in 1998 [41], and Saroiu et al. analyzed the Web traces of University of Washington four years later [31], both of them focus on the general characteristics of HTTP traffic, especially on static Web pages. Although uncacheable content are also discussed in [41], the work presented in this paper is more thorough. Fu et al. focused their work on reconstructing HTTP request/response, logically grouping Web objects belonging into to one logic Web page together, virtually rebuilding the Web page and monitoring Internet service performance at a server site [15]. Feldmann's work is a perfect online monitoring tool, but just as the works mentioned above, they were all only interest in the HTTP protocol information appeared in HTTP header part. Our work reconstructed the response Web objects and calculate the content digest based on the rebuilt response objects and make use of object digest repeatness to investigate possibility of reusing.

Kelly and Mogul [23]instrumented a non-caching, cache busting proxy to collect response trace, and analyzed URL-alias based on response object digest. Our work is also based on the object digest, but emphasize on the cacheability of uncacheable content. Leibowitz et al. [26] investigated P2P traffic at fixed destination port number, and concluded that P2P traffic is highly repetitive and consequently responds well to caching. Our work investigate all possible destination port number, and find that the well-known port only took 13% of KaZaA traffic.

Finally, our work is also related to several previous studies of Web server performance [2, 18, 20, 24, 28, 35, 39], but differs from them in that the overhead of dynamic Web content generation and related network transfer time are studied. Although closest in spirit, the work by Brewington and Cybenko [8] and Douglis et al. [11] on understanding the dynamics of the Web differs from our efforts in that only uncacheable Web content are studied in our analysis, while the Web content analyzed in their work has broader characteristics, with a large fraction actually corresponding to static content.

7 Summary and Future Work

In this paper, we have collected and analyzed an one-day period HTTP traffic passing over a medium-size educational institution, emphasized on the workload characterization of uncacheable web content. Through our observation, we inferred four promising directions to improve caching and content delivery: first, pushing the functionality of uncacheable content generation to the network edges; second, applying the access pattern feature to prefetching scheme; third, implementing an efficient content-based P2P traffic caching; fourth, combining digest-based approach

into current cache to exploit the prevailing URL-alias phenomenon. Our future work will focus on exploiting these opportunities and integrating them into our ongoing CONCA project.

References

- [1] Akamai Technologies Inc., <http://www.akamai.com/>.
- [2] V. A. Almeida and M. A. Mendes. Analyzing the impact of dynamic pages on the performance of web servers. *Proceedings of the Computer Measurement Group Conference*, Dec. 1998.
- [3] M. Arlitt and C. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking* 5(5):631–645, 1997.
- [4] A. Awadallah and M. Rosenblum. The vMatrix: A network of virtual machine monitors for dynamic content distribution. *Proc. of the 7th International Workshop on Web Caching and Content Distribution (WCW'02)*, Aug. 2002.
- [5] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella. Changes in web client access patterns: Characteristics and caching implications. *World Wide Web, Special Issue on Characterization and Performance Evaluation* 2:15–28, 1999, <http://www.cs.bu.edu/techreports/1998-023-web-client-trace-changes-and-implications.ps.Z>.
- [6] P. Barford and M. E. Crovella. Generating representative web workloads for network and server performance evaluation. *Proceedings of Performance '98/ACM SIGMETRICS '98*, July 1998.
- [7] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. *Proc. of IEEE Conference on Computer Communications (INFOCOM'99)*, pp. 126-134, Mar. 1999, <http://www.research.att.com/~breslau/papers/zipf.ps.gz>.
- [8] B. E. Brewington and G. Cybenko. How dynamic is the web? *Proc. of the 9th International World Wide Web Conference (2000)*, May 2000.
- [9] P. Cao, J. Zhang, and K. Beach. Active cache: Caching dynamic contents on the web. *Proc. of IFIP Int'l Conf. Dist. Sys. Platforms and Open Dist. Processing*, pp. 373-388, 1998, <http://www.cs.wisc.edu/~cao/papers/active-cache.ps>.
- [10] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking* 5(6):835–846, 1997, <http://www.cs.bu.edu/fac/best/res/papers/ton97.ps>.
- [11] F. Douglass, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: a live study of the world wide web. *Proc. of the 1st USENIX Symposium on Internet Technologies and Systems (USITS'97)*, pp. 147-158, Dec. 1997, <http://www.douglass.org/fred/work/papers/roc.pdf>.
- [12] A. Feldmann. BLT: Bi-layer tracing of http and tcp/ip. *Proc. of the 9th International World Wide Web Conference (2000)*, May 2000.
- [13] A. Feldmann, R. Caceres, F. Douglass, G. Glass, and M. Rabinovich. Performance of web proxy caching in heterogeneous bandwidth environments. *Proc. of IEEE Conference on Computer Communications (INFOCOM'99)*, pp. 107-116, Mar. 1999, <http://www.douglass.org/fred/work/papers/hetproxcache.pdf>.
- [14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext transfer protocol, HTTP/1.1, 1999, <http://www.ietf.org/rfc/rfc2616.txt>.
- [15] Y. Fu, L. Cherkasova, W. Tang, and A. Vahdat. EtE: Passive end-to-end internet service performance monitoring. *Proceedings of the 2002 USENIX Annual Technical Conference*, June 2002.
- [16] Gnutella, <http://gnutella.wego.com>.
- [17] S. D. Gribble and E. A. Brewer. System design issues for internet middleware services: Deductions from a large client trace. *Proc. of the 1st USENIX Symposium on Internet Technologies and Systems (USITS'97)*, Dec. 1997.
- [18] Y. Hu, A. Nanda, and Q. Yang. Measurement, analysis and performance improvement of apache web server. *Proceedings of the IEEE International Performance, Computing, and Communications Conference*, Feb. 1999.
- [19] IBM Corp. Websphere platform, <http://www.ibm.com/websphere>.
- [20] A. Iyengar, J. Challenger, D. Dias, and P. Dantzig. High-performance web site design techniques. *IEEE Internet Computing* 4(2), March/April 2000.
- [21] KaZaA, <http://www.kazaa.com>.
- [22] T. Kelly. Thin-client web access patterns: Measurements for a cache busting proxy. *Proc. of the 6th International Workshop on Web Caching and Content Distribution (WCW'01)*, June 2001, http://www.cs.bu.edu/techreports/2001-017-wcw01-proceedings/129_beck.pdf.

- [23] T. Kelly and J. Mogul. Aliasing on the world wide web: Prevalence and performance implications. *Proc. of the 11th International World Wide Web Conference (2002)*, May 2002.
- [24] B. Kothari and M. Claypool. Performance analysis of dynamic web page generation technologies. *Proceedings of the International Network Conference (INC)*, July 2000.
- [25] B. Krishnamurthy and J. Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measurement*. Addison-Wesley, Inc, 2001.
- [26] N. Leibowitz, A. Bergman, R. Shaul, and A. Shavit. Are file swapping networks cacheable? characterizing p2p traffic. *Proc. of the 7th International Workshop on Web Caching and Content Distribution (WCW'02)*, Aug. 2002.
- [27] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: Findings and implications. *Proc. of ACM SIGCOMM'00*, Aug. 2000, <http://research.microsoft.com/~liliq/papers/pub/sigcomm2000.ps>.
- [28] V. Pai, P. Druschel, and W. Zwaenepoel. Flash: An efficient and portable web server. *Proceedings of the 1999 USENIX Annual Technical Conference*, June 1999.
- [29] M. Rabinovich, Z. Xiao, F. Douglis, and C. Kamanek. Moving edge side includes to the real edge – the clients. *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Mar. 2003.
- [30] J. Ravi, W. Shi, and C. Xu. Pace: Prefetching and filtering of personalized emails at the network edges. Tech. Rep. CS-MIST-TR-2003-005, Department of Computer Science, Wayne State University, May 2003.
- [31] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. *Proc. of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, Dec. 2002.
- [32] W. Shi, E. Collins, and V. Karamcheti. DYCE: A synthetic dynamic web content emulator. *Poster Proc. of 11th International World Wide Web Conference*, May 2002, <http://www.cs.wayne.edu/~weisong/papers/dyce.pdf>.
- [33] W. Shi, E. Collins, and V. Karamcheti. Modeling object characteristics of dynamic web content. *Journal of Parallel and Distributed Computing (to appear)*, Sept. 2003.
- [34] W. Shi and V. Karamcheti. CONCA: An architecture for consistent nomadic content access. *Workshop on Cache, Coherence, and Consistency(WC3'01)*, June 2001.
- [35] W. Shi, R. Wright, E. Collins, and V. Karamcheti. Workload characterization of a personalized web site — and its implication on dynamic content caching. *Proc. of the 7th International Workshop on Web Caching and Content Distribution (WCW'02)*, pp. 1-16, Aug. 2002, <http://www.cs.wayne.edu/~weisong/papers/wcw02.pdf>.
- [36] Squid web cache, <http://www.squid-cache.com/>.
- [37] tcpdump, <http://www.tcpdump.org>.
- [38] M. Tsimelzon, B. Weihl, and L. Jacobs. ESI language sepcification 1.0, 2000, <http://www.esi.org>.
- [39] M. Welsh, D. Culler, and E. Brewer. SEDA: An architecture for well-conditioned, scalable internet services. *Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP-18)*, Oct. 2001, <http://www.cs.berkeley.edu/~mdw/papers/seda-sosp01.pdf>.
- [40] D. Wessels. *Web Caching*. O'Reilly Inc., 2001.
- [41] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. M. Levy. Organization-based analysis of web-object sharing and caching. *Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems (USITS'99)*, 1999, <http://www.cs.washington.edu/research/networking/websys/pubs/usits99.ps%>.
- [42] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. *Proc. of 17th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 16-31, Dec. 1999.