Workload Characterization of Uncacheable HTTP Traffic

Zhaoming Zhu, Yonggen Mao, and Weisong Shi
{zhaoming,ygmao,weisong}@wayne.edu

Technical Report: **MIST-TR-03-003**

June 2003

Mobile and Internet SysTem Group (MIST)
Department of Computer Science
Wayne State University
Detroit, MI 48202
http://mist.cs.wayne.edu

Workload Characterization of Uncacheable HTTP Content

Zhaoming Zhu, Yonggen Mao, and Weisong Shi

Department of Computer Science
Wayne State University, Detroit, Michigan

Abstract. We have witnessed the inefficiency of traditional proxy caching due to the rapid growth of uncacheable HTTP content, resulting from ever-increasing dynamic Web services, cache busting technologies, and the emergency of various HTTP-based applications. Although several approaches have been proposed for caching fragment-based dynamic Web content, we believe that a general understanding of the characteristics of uncacheable HTTP content is of great importance to the future of HTTP-based content caching and delivery.
In this paper, we characterized the uncacheable HTTP traffic and found that (1) compared with cacheable content, uncacheable content now dominates the whole HTTP traffic (96.5%), and surprisingly, multimedia type content transferred by P2P applications (35.5%) and graphic type (jpeg and gif) content (17.3%) are top two in the uncacheable HTTP objects; (2) although dynamic content generation consumes longer server processing time, network latency dominates the total user-perceived latency for dynamic Web content; (3) a considerable (50%) portion of content has their TTL value equals to zero. Among these content, we observed URL-alias contributes 20% of total number of responses; and (4) P2P traffic is increasingly scattered among non-default tcp ports which is a big challenge for P2P content caching. Furthermore, several implications derived from these observations are discussed as well as future directions for efficient content caching and delivery of uncacheable content are proposed .

1 Introduction

Proxy 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 the increasingly large portion of today's HTTP traffic: *the uncacheable, probably dynamically generated HTTP objects*, including dynamic generated and personalized Web content, and fast growing peer-to-peer file sharing traffic, which is not suitable for traditional proxy caching due to its "fetch-at-most-once" behavior property [1]. To face the rapid growth trend of uncacheable content in the HTTP traffic, HTTP content with uncacheable property deserves 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*. The success of these techniques depends on the understanding of the HTTP uncacheable content, including content characteristics and access patterns. In this paper, we tried to

answer these following questions: Among the huge HTTP content delivered on the Internet, what part are cacheable and what part are 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 HTTP content.

To answer these questions, we sniff and analyze all inbound and outbound HTTP traffic on all possible TCP ports, at Wayne State University (WSU), a medium-sized educational institution with 35,000 students, faculty and staff. The *tcpdump* [2] 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 connections. By analyzing a one-day trace [1], we observed the following: (1) uncacheable data have dominated the majority portion of today's HTTP traffic (96.5% of total transferred HTTP content), and surprisingly, multimedia type content transferred by P2P applications (35.5%) and graphic type (jpeg and gif) content (17.3%) are top two in the uncacheable HTTP objects; (2) compared to cacheable content, uncacheable content consumes 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 (13 K bytes vs. 7K bytes); (4) clients that accessing personalized content and servers that providing 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 [3, 4]; (5) a considerable (50%) portion of HTTP content has their TTL values equal to zero. Among these content, we observed URL-alias contributes 20% of total requests; (6) P2P traffic is increasingly scattered among multiple ports (only 13% from default ports for KaZaA [5] traffic), which is a big challenge for deployment of P2P traffic caching. Several implications could be derived based on above observations: (1) a considerable portion of uncacheable HTTP content is cacheable; (2) domination of network latency factor motivates the moving of functionality for uncacheable HTTP content generation to the edge of network; (3) prefetching for personalized Web content is promising because of the concentrated popularity of clients and servers; (4) convinced by the observed P2P request popularity, we believe that content-based caching is significant to the ever-increasing P2P traffic; (5) exploiting URL-alias is a promising direction to improve cacheability of uncacheable content.

The rest of this paper is organized as follows. The following section introduces content classification techniques for uncacheable content. Section 3 describes the methodology used in our study. Section 4 starts with a high level characteristics analysis, followed by detail discussion of the cacheable and uncacheable content traffic patterns and meaningful features. Several implications are discussed in Section 5. Finally, related work and summary are listed in Section 6 and Section 7 respectively.

[1] Later in Section 4 we will explain that why do we believe one day trace is enough for this analysis.

2 Background

From the viewpoint of proxy caching, generally HTTP object could be broadly categorized as uncacheable content or cacheable content. The cacheable HTTP objects refer to those infrequently changed HTTP objects (also known as static HTTP content).

Based on the HTTP protocol specifications [6], the uncacheable HTTP content could be further classified into seven uncacheable subtypes, depends on the following criteria:

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

Subtype 2 - `DynGen`: If the method is `GET`, and the request URL contains keywords (like "cgi", "asp", "=" and "?", ...etc.), which implies the HTTP response object is probably generated dynamically, then that 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 object could be considered as `Pragma` subtype;

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

Subtype 5 - `Personalized`: If the HTTP request header contains `Cookie` or Authorization related headers,or the HTTP response contains `Set-Cookie` header, the corresponding HTTP content is defined as `personalized` subtype;

Subtype 6 - `AbnormalStatus`: If the return status code, from server, does not belong to 2XX or 3XX, we think the response object is not a cacheable response and treat it as of `AbnormalStatus` subtype;

Subtype 7 - `ZeroTTL`: Except above six subtypes, we are also interested in the HTTP objects whose TTL (time-to-live) value equals to zero. This sort of objects 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 possible HTTP traffic, TCP packets on all ports are sniffed.

To extract the complete HTTP information, including both header and content, we have developed WebTACT, an **W**eb **T**raffic **A**nalysis and **C**haracterize **T**ool. WebTACT consists of three steps, as shown in Figure 1. In the first *TCP connection reconstruction* step, TCP sessions are reconstructed. Our algorithm is similar to that proposed in [7]. One TCP connection is identified by the first several TCP packets(SYN, SYN/ACK, ACK), used in 3-way hand-shaking period. Followed TCP packets are determined to belong this connection, based on their sourec/destionation IP address and port numbers combination, and whether their captured time is within the 2MSL (Maximum Segment Lifetime, we choose 60 seconds in our analysis) time interval after receiving the latest

packet from this connection. After the TCP connection is rebuilt, all TCP packets payload are accumulated to rebuild the buffer content, in the *HTTP pair reconstruction* step. Finally, in the *HTTP pair data extraction* step, we extract the interesting information from the corresponding HTTP headers, calculate the hash digest values for the `Cookie` field and the HTTP object, 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. It is not unusual to see multiple pairs along with one persistent connection, introduced in HTTP 1.1 protocol [6]. In this situation, based on the assumption that the order of multiple requests/response are matched perfectly [8], we consider 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).

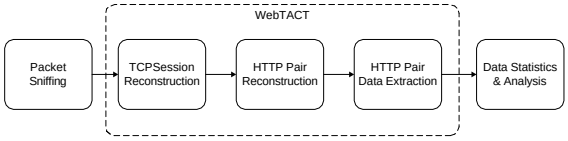


Fig. 1. A diagram of data collection and reconstruction.

TTL (time-to-live) is defined as the difference between its 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 for which is specified by the `Age` header. Our calculation of freshness time is based on the same heuristic algorithm used in Squid Web cache [9].

4 Analysis Results

We collected one-day period (12:00 pm, Mar 18 -12:00pm Mar 19, 2003) HTTP traffic, rebuilt and investigated the contained HTTP traffic. We believe that one-day trace is enough for our analysis, because of the following reasons: (1) the diary-based access pattern has been observed in several previous analysis [10]; (2) unlike previous efforts which look at only HTTP header information, we are interested in both headers and the real content, which will consume a lot of disk space. Analysis results and observations are depicted 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 objects are targeted to clients inside WSU campus, while the outbound traffic means that the response objects are targeted to clients outside WSU campus. The total distinct client number inside WSU campus is 9,053, and that outside WSU is 93,250. The total server (host providing HTTP content) number inside WSU

is 1,930, and that outside WSU is 114,416. 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` [11].

From Table 1, we can see that, the captured-reconstructed gross HTTP traffic (include HTTP headers and bodies) is around 126G bytes. For the total objects size (or the total size of transferred HTTP response objects),[2] the uncacheable content outnumbers the cacheable content (77G Bytes vs. 2.8 G Bytes). The servers that providing uncacheable content outnumber those providing cacheable content (116,149 vs. 7,518), while the clients accessing dynamic content also largely outnumber the clients accessing cacheable content(101,971 vs. 14,674). These data exemplify that the uncacheable content dominates today's HTTP traffic portion. Figure 2 gives out the top 15 uncacheable content types, in total bytes and request/response numbers, and in both traffic directions. As shown in Figure 2(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) 99.6% KaZaA HTTP objects are categorized into uncached type by our analyzer.

The reason for the large percentage of "Unknown content type" in Figure 2(b) is that a large number of responses (3,168,821, 35.7% of total 8,869,630 uncacheable responses) are with zero-size object, 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 [4], 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 possible reason is widely acceptable of cache busting technologies [8]. The multimedia type objects (`video/x-msvideo`, `video/mpeg` and `audio/mpeg`), which contribute to a large percentage of total bytes and a small percentage of total number of responses, implies a larger average size of these kinds of objects.

Type	Cacheable		Uncacheable	
HTTP Traffic Direction	Inbound	Outbound	Inbound	Outbound
# of Servers	7,345	173	114,221	1928
# of Clients	7,007	7,667	9,050	92,921
Total Gross Traffic(bytes)	2,282,983,567	917,770,363	69,497,564,141	53,495,360,620
Total Object Size (bytes)	2,010,062,553	865,105,662	44,259,882,154	33,155,217,994
# of Requests	309,616	73,662	6,742,014	2,234,888

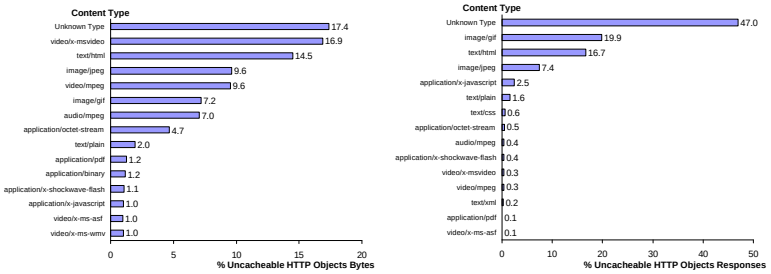
Table 1. High-level statistics of HTTP traffic.

4.2 Detailed Characteristics of Uncacheable HTTP Content

After describing the breakdown of different uncacheable subtypes first in this section, we in turn present the response time, object size distribution, popularity and access pattern, P2P traffic, and other interesting features of uncacheable content as follows. Due to space limitation, we present, in another technical report version of this paper, the analysis results about time-relating access pattern and its implication.

[2] Hereafter, all the traffic mentioned in this paper are referring to total objects size.

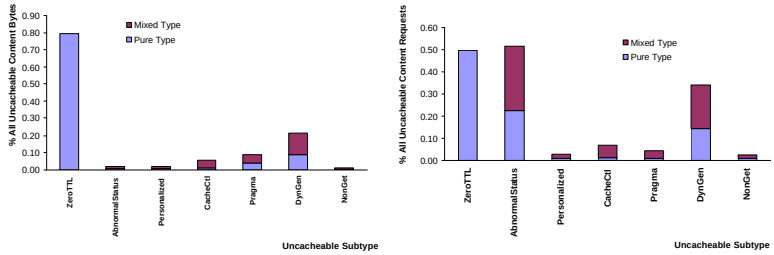
Uncacheable Content Breakdown Figure 3 shows the percentage breakdown for each of the seven uncacheable subtypes, by their total object size and request/response number, and Table 2 lists the detail absolute numbers. The "mixed" type means the uncacheable subtype is a combination of this subtype and at least one other uncacheable subtype, while the "pure" type means the request belongs to this subtype only, not combined with other subtypes. From Figure 3, we found that the `personalized` objects (subtype 5) consists of less than ten percent of all uncacheable content in terms of both bytes and number of requests, not as large as previous observation [7]. We do not know the exact reason for this low percentage of personalized HTTP objects.



(a) (b)

Fig. 2. Histogram of top uncacheable HTTP content type by traffic bytes and responses.

A distinguish portion, `ZeroTTL` (subtype 7), implies a promising probability of caching performance improvement that we will give more detail analysis later.



(a) (b)

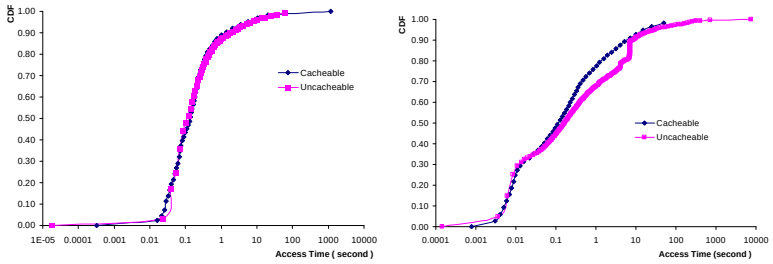
Fig. 3. Histogram of all 7 uncacheable subtypes in terms of object sizes and the number of requests.

Subtype	ZeroTTL	AbnormalStatus	Personalized	CacheCtl	Pragma	DynGen	NonGet
pure requests	4,413,886	2,067,218	71,634	104,186	80,947	1,274,334	69,767
pure bytes	61,009,017,498	1,223,834,086	576,377,034	1,021,726,984	3,067,833,909	6,808,561,668	289,161,943
mixed requests	0	574,645	92,989	402,960	232,048	476,420	80,710
mixed bytes	0	365,421,902	416,873,201	2,474,101,783	875,652,699	2,956,389,265	369,123,078

Table 2. Detail breakdown for all seven uncacheable subtypes.

Response Time and Breakdown for Uncacheable Content For further analysis, we first want to know, whether the cacheability of 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 got from conventional server or proxy logs.



(a) Inbound direction (b) Outbound direction

Fig. 4. CDF of response time for cacheable/uncacheable objects with different traffic direction.

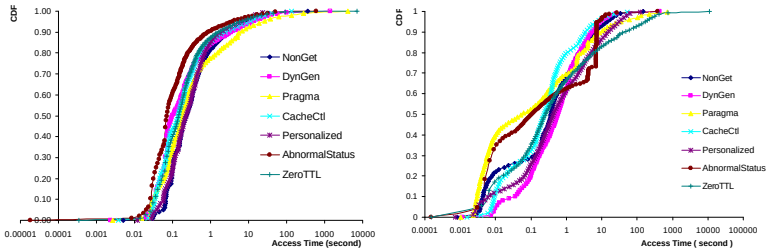
Figure 4 shows the cumulative distribution function (CDF) of response time for cacheable/uncacheable objects, in both inbound and outbound directions. In this figure, the x-axis represents the response time in ascending order, and the y-axis represents the cumulative percentage of responses for the corresponding x-axis value. Because of the sniffing point location 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 inbound HTTP traffic, the difference between the response time of uncacheable and cacheable objects is not so distinguish. This implies that the time difference caused by dynamic/static content generation has been blurred by the network latency on their route.

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

Figure 5 shows the CDFs of response time for the six subtypes uncacheable objects. Pure subtype data sources are applied to avoid interference from other subtypes. From this figure we can see the response times for different dynamic type do not show much difference, especially for inbound traffic. The large size P2P (KaZaA) objects, which take longer time period to finsih the HTTP session and are categorized mainly in `Personalized` and `ZeroTTL` subtypes, only occupy a very small count percentage and do not affect the CDF curve much.

Object Size Distribution Object size distribution is also an interesting topic of our study, especially when HTTP objects are classified into two major classes: cacheable and uncacheable. Figure 6 shows the CDF for object size distribution. For uncacheable objects, a large portion (37%) of objects'size is zero, so we exclude these zero-size ob-



(a) Inbound direction (b) Outbound direction

Fig. 5. CDF of response time for seven uncacheable subtypes objects in both directions.

jects in the data source of Figure 6. 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 6 shows that 90% of cacheable objects smaller than 14,860 bytes, while same percentage uncacheable objects are smaller than 18,694 bytes. The average size is 7K bytes (cacheable) vs. 13K bytes (uncacheable).

Amazingly, the largest HTTP object size we observed is 252 M bytes for uncacheable object and 12M bytes for cacheable objects. These numbers are much smaller than that appear in [11]. The possible reasons are: (1) our data collecting period is relatively short (24 hours vs. 9 days data collecting period [11]); (2) the object size is calculated based on the bytes on the wire, instead of the HTTP headers. As more and more applications (e.g., KaZaA) adopt parallel downloading or other segment-based content delivery techniques, supported by the HTTP protocol, we believe the size of individual HTTP objects will not be larger any more. So the real reconstructed (fragmented) objects reflecting only a fraction of total size is a reasonable explanation.

Is Uncacheable Content Really Uncacheable? Although the object composition technique, such as ESI [12], has been proposed, in this paper we are looking for URL-alias derived cacheable possibility. There exist a large number of HTTP 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 7 shows the CDF of ZeroTTL objects digest, based on the rank of the number of repeatness of the same digest value, not on their URLs. Totally, there are 4,413,886 objects belonging to ZeroTTL subtype. Among these, there are almost 21% (943,476) are zero-size objects, that are excluded in the data source of Figure 7. In Figure 7, we observe that many different URLs share the identical content digest. This is caused by the phenomenon called “URL-alias” [13]. The figure shows that the 1st rank repeated digest value repeats 16,918 times, the 100th repeats 971 times, and the 1000th repeats 167 times. The number of requests targeting the top 1000 (less than 0.1% of total distinct digest value) rank digest value count for 18% 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.

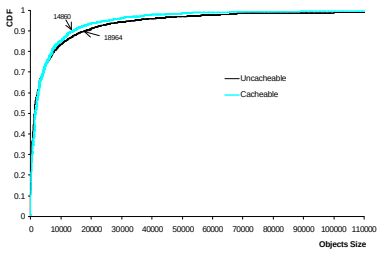


Fig. 6. Size distribution for cacheable / uncacheable objects.

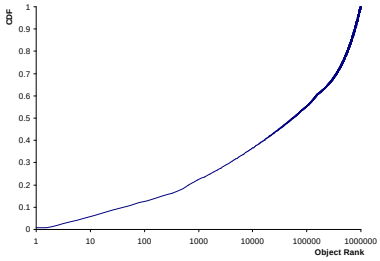
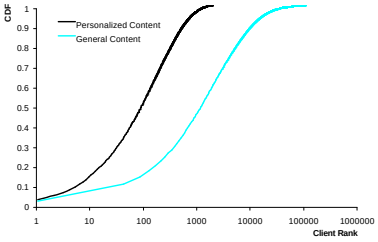
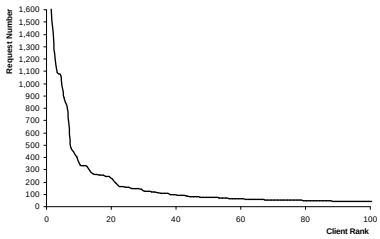


Fig. 7. CDF of repeated digest for ZeroTTL objects.

Popularity Analysis Client/Server Popularity Figure 8 and 9 plot the client/server popularity, when accessing/providing personalized content and general HTTP content. We assume that the personalized Web content would be more client/server-specific, than general content, due to its "personalized" property. And these figures do verify our assumption.



(a) Client popularity

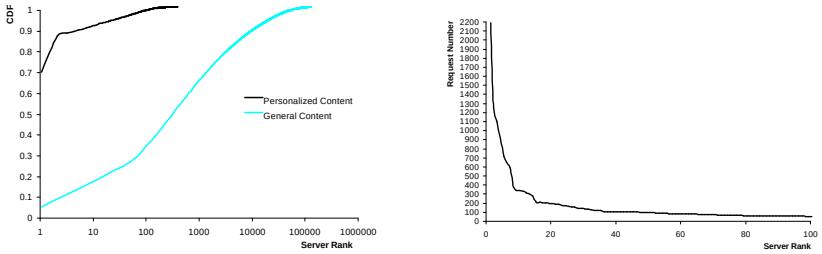


(b) Top 100 rank inbound clients distribution

Fig. 8. Clients popularity for general content and personalized content.

Figure 8(a) shows that clients consuming personalized content are more concentrated than the general clients population. Top 1% of clients that accessing personalized content bring about 20% of the total personalized content requests. However, unlike previous observations [3, 4], we find that clients interested in personalized content only occupy 2% of the total client population. In Figure 8(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 9(a) also shows that personalized content is provided by 1% 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 85% of the requests for personalized content requests. In Figure 9(b) we plot the request distribution of the top 100 servers. The graph shows the existence of the "hot" personalized Web servers and the top 30 of the servers contribute 95% of the total requests among the top 100 servers providing personalized content.



(a) Server popularity (b) Top 100 rank inbound servers distribution

Fig. 9. Servers popularity for general content and personalized content.

Object Popularity Due to the personalized property, personalized HTTP objects might not be more concentrated than general objects. Figure 10 plots the object popularity for general content and personalized content, and verifies our assumption. It also shows that personalized content only occupy less than 10% of the total number of requests, as already shown in Figure 3 and Table 2.

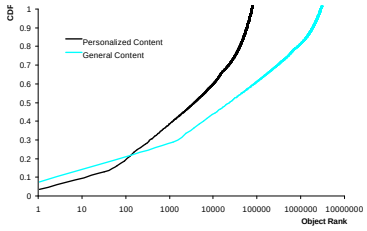


Fig. 10. HTTP objects popularity.

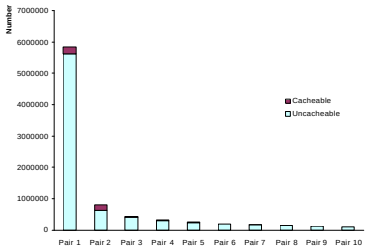


Fig. 11. Uncacheable/cacheable distribution in persistent connection.

Persistent Connection vs. Uncacheable HTTP Objects In our collected-reconstructed HTTP data, there are totally 669,958 persistent connection sessions, consists of 3,411,741 HTTP request/response pairs. On average, a persistent session consists of 5.09 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. If such distribution pattern does exist, then in Figure 11, the height of the columns cooresponding to pair 2, 3 and so on, should be match to that of pair 1. But we did not observe that. From Figure 11, we also could conclude that most of the uncacheable content does not appear in persistent connections.

Peer-to-Peer Traffic Analysis We reconstructed all http-based P2P traffic by capturing all TCP traffic, instead of sniffing only some specific default ports (e.g., 1214 for KaZaA, 6346 and 6347 for Gnutella, used by previous work [11]). Generally, as ob-

served earlier in this Section, P2P traffic contributes to a large portion of total HTTP traffic.

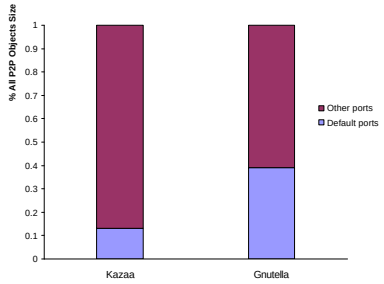


Fig. 12. HTTP objects size by P2P applications using different ports

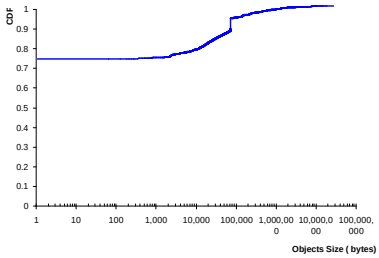


Fig. 13. CDF of KaZaA object size distribution

For Gnutella type P2P applications [14], we find several different Gnutella client applications (e.g., `LimeWire`, `BearShare`, `Shareaza` etc.) 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 HTTP object size transferred by Gnutella applications is 1,110,383,667 bytes, while that by KaZaA application is 26,659,686,069 bytes. The total object size transferred by P2P applications occupies 33.8.5% of the total observed HTTP object size while the corresponding percentage is over 75% in [11]'s work.

Excluding zero-size object, the average object size of P2P objects is 171K bytes, while that of WWW(without P2P) objects is 13K bytes. On average, P2P object size is one order of magnitude larger than that of WWW (without P2P) objects, while [11] 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 [6].

Figure 12 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. Figure 13 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 [11]. 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 5.4% of total objects), which corresponds size around 64 K bytes. This phenomenon strongly implies a prevailing partial object transmission happened in P2P application traffic, which applying IP packet's maximum packet total length.

5 Implications for Caching and Delivery of Uncacheable Content

The analysis of WSU HTTP traffic indicates both the need to improve delivery of uncacheable HTTP 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 HTTP content becoming an important part of current-day HTTP traffic. Our study has shown that 96.5% of HTTP traffic is uncacheable, including peer-to-peer traffic. Unfortunately, traditional proxy caching and CDNs developed to improve delivery static content do not yield the same benefits for these uncacheable content. This situation will lead client-perceived latencies increased and lacking of network bandwidth of various network applications. Some initial results on caching of peer-to-peer traffic are provided in [1].
– **Is uncacheable really uncacheable?** Although we observed most of HTTP traffic was uncacheable content, but 38% 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 [12], we believe a considerable part of uncacheable content is cacheable.
– **Need for migrating the dynamic 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 needs more server processing time for its generation than cacheable content. The possible reason is that the network latency has blurred the difference. This implies that the further server-side effects will not 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 [15].
– **Prefetching for personalized content** Clients that accessing 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.
– **Potential for P2P traffic caching** The P2P traffic accounts for 33.8% objects bytes of total HTTP 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 `Personalized` and `ZeroTTL` subtypes 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, in spite of the fact that individual user will access the content only once [1]. However P2P traffic is increasingly scattered among multiple ports (only 13% from default ports for KaZaA [5] traffic), which is a challenge for real deployment of P2P traffic caching.

6 Related Work

Web workload characterization has been extensively studied in the past [8]. 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 to previous work on understanding the characteristics of dynamic and personalized Web content [16] and peer-to-peer traffic [17, 1].

The methodology used in our analysis is very close to that of previous work [3, 18, 11, 4], 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 [4], and Saroiu et al. analyzed the Web traces of University of Washington four years later [11], both of them focus on the general characteristics of HTTP traffic, especially on static Web pages. Although uncacheable content are also discussed in [4], the work presented in this paper is more thoroughly. 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 [18]. 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 [13]instrumented a non-caching, cache busting proxy to collect response trace, and analyzed URL-alias based on response object digest. Our work also based on the object digest, but emphasized on the cacheability of uncacheable content. Leibowitz et al. [17] investigated P2P traffic at fixed destination port number, and concluded that P2P traffic is highly repetitive and consequently responds well to caching. Our work investigated all possible destination port number, and found that the well-known port only took 13% of KaZaA traffic.

Although closest in spirit, the work by Brewington and Cybenko [19] and Douglis et al. [20] 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; Finally 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 [21].

References

1. Gummadi, K., Dunn, R., Gribble, S., Levy, H.: Measurement, modeling, and analysis of a peer-to-peer file sharing workload. In: Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP-19). (2003)
2. (tcpdump), `http://www.tcpdump.org`
3. Feldmann, A., Caceres, R., Douglis, F., Glass, G., Rabinovich, M.: Performance of web proxy caching in heterogeneous bandwidth environments. In: Proc. of IEEE Conference on Computer Communications (INFOCOM'99). (1999) 107–116
4. Wolman, A., Voelker, G.M., Sharma, N., Cardwell, N., Brown, M., Landray, T., Pinnel, D., Karlin, A., Levy, H.M.: Organization-based analysis of web-object sharing and caching. In: Proc. of USITS'99. (1999)
5. (KaZaA), `http://www.kazaa.com`
6. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Bernes-Lee, T.: RFC 2616: Hypertext transfer protocol, HTTP/1.1 (1999)
7. Feldmann, A.: BLT: Bi-layer tracing of http and tcp/ip. In: Proc. of the 9th International World Wide Web Conference (2000). (2000)
8. Krishnamurthy, B., Rexford, J.: Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measurement. Addison-Wesley, Inc (2001)
9. (Squid web cache), `http://www.squid-cache.com/`
10. Shi, W., Collins, E., Karamcheti, V.: DYCE: A synthetic dynamic web content emulator. In: Poster Proc. of 11th International World Wide Web Conference. (2002)
11. Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., Levy, H.M.: An analysis of internet content delivery systems. In: Proc. of the Fifth USENIX Symposium on Operating Systems Design and Implementation. (2002)
12. Tsimelzon, M., Weihl, B., Jacobs, L.: ESI language sepcification 1.0 (2000)
13. Kelly, T., Mogul, J.: Aliasing on the world wide web: Prevalence and performance implications. In: Proc. of the 11th International World Wide Web Conference (2002). (2002)
14. (Gnutella), `http://gnutella.wego.com`
15. Ravi, J., Shi, W., Xu, C.: Pace: Prefetching and filtering of personalized emails at the network edges. Technical Report MIST-TR-2003-005, Department of Computer Science, Wayne State University (2003)
16. Shi, W., Wright, R., Collins, E., Karamcheti, V.: Workload characterization of a personalized web site — and it's implication on dynamic content caching. In: Proc. of the 7th International Workshop on Web Caching and Content Distribution (WCW'02). (2002) 1–16
17. Leibowitz, N., Bergman, A., Shaul, R., Shavit, A.: Are file swapping networks cacheable? characterizing p2p traffic. In: Proc. of the 7th International Workshop on Web Caching and Content Distribution (WCW'02). (2002)
18. Fu, Y., Cherkasova, L., Tang, W., Vahdat, A.: EtE: Passive end-to-end internet service performance monitoring. In: Proceedings of the 2002 USENIX Annual Technical Conference. (2002)
19. Brewington, B.E., Cybenko, G.: How dynamic is the web? In: Proc. of the 9th International World Wide Web Conference (2000). (2000)
20. Douglis, F., Feldmann, A., Krishnamurthy, B., Mogul, J.: Rate of change and other metrics: a live study of the world wide web. In: Proc. of the 1st USENIX Symposium on Internet Technologies and Systems (USITS'97). (1997) 147–158
21. Shi, W., Karamcheti, V.: CONCA: An architecture for consistent nomadic content access. In: Workshop on Cache, Coherence, and Consistency(WC3'01). (2001)