A rationale for web caching in consumer ISPs: The impact of DNS lookup times and HTTP session characteristics

Jason But, Urs Keller[1], Grenville Armitage
Centre for Advanced Interent Architectures. Technical Report 050218A
Swinburne University of Technology
Melbourne, Australia
urs.keller@epfl.ch, jbut@swin.edu.au, garmitage@swin.edu.au

Abstract– Web caches are generally considered a useful tool because they reduce replication of network traffic flowing from original content sources. In this paper we experimentally characterise the network and transport layer consequences of web caching in the consumer ISP context. We instrumented a small number of Australian, broadband-attached homes to collect round-trip time (RTT) and hop count statistics for their HTTP/TCP sessions, and collect DNS lookup statistics associated with each HTTP exchange. We estimated the impact of DNS lookup delays on overall HTTP session times, and use our RTT and hop count statistics to show that consumer ISPs would benefit greatly from local caching, particularly in Australia where speed of light delays have a large impact on sessions times when retrieving international content.

Keywords– Measurement, Performance, Experimentation, Verification

I. Introduction

As the Internet evolves it is increasingly being used for the delivery of both entertainment and information services. The growing deployment of cost-effective, consumer-focused broadband access technologies ensures that the Internet will be the platform of choice for content delivery and entertainment applications. These applications include online multiplayer gaming, Internet personal communications, and commercially-viable streaming audio and video applications. Internet service providers (ISPs) constantly face the challenge of ensuring their network resources will handle growth in demand driven by new applications and growing customer base.

One technique that tempers the demand for core network resources is the caching of content at points closer to the content consumers. Caching puts copies of content closer to end-users, thus reducing the need for original content to be repeatedly downloaded from (possibly remote) content servers. This helps reduce the replication of traffic across network links. However, although the benefits of caching to the network core are generally accepted, it remains an open question as to the impact of caching on the network's performance as perceived by the end-user.

In this paper we explore data sets of Internet usage gathered at three domestic residences over a period of approximately three months, and quantify the perceived performance of the web-based applications being run by the test subjects. We gathered round-trip time (RTT) and hop count statistics for their HTTP/TCP sessions, and collect DNS lookup statistics associated with each HTTP exchange. As shown in Figure 1, the overall time taken to service a typical HTTP request can include a preceding DNS resolution request. For this reason we used our raw data to estimate the impact of DNS lookup delays on overall HTTP session times, and used our RTT and hop count statistics to show that consumer ISPs would benefit greatly from local caching (particularly in Australia where speed of light delays have a large impact on session times when retrieving international content).

The paper is organised as follows: in Section 2 we explain our experimental setup and how our data is collected. Section 3 discusses our experimental results, while Section 4 extends the meaning of these results to other online applications.

II. Experimental Methodology

Our data collection testbed consists of three home Internet sites, each configured with a 512kbps downstream/128kbps upstream ADSL connection. These sites are located in Melbourne, Australia. Each site has been provided with a FreeBSD gateway which manages an ADSL modem in bridged Ethernet mode, see Figure 2. The gateway:

- Reconnects to the ISP if the connection is dropped.
- Provides a DHCP service to any computer within the home LAN.
- Provides a non-cached DNS forwarding service within the home LAN.
- Provides a Network Address Translation (NAT) service to allow Internet connection sharing for any computer within the home LAN.
- Restarts the data gathering application upon failure.

Data gathering is performed using the **netsniff** application [8] installed on the FreeBSD gateway. All network traffic is anonymised and analysed as per the applications

[1]Urs Keller was a visitor at CAIA from the Swiss Federal Institute in Lausanne $EPFL$.

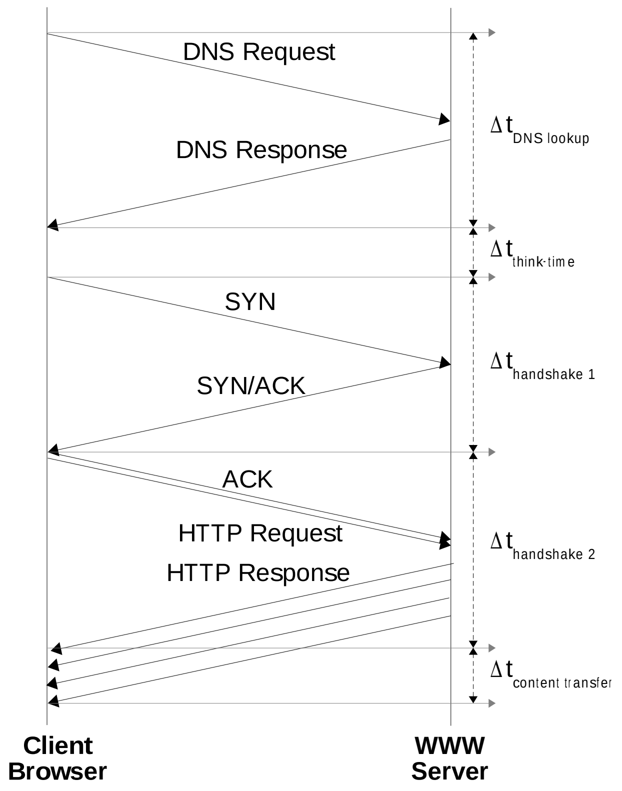

Figure 1: Time diagram of HTTP connections

capabilities [8]. The logs are rolled over daily and transmitted to a central server for later analysis:

- Log rollover and transfer is scheduled for 3:00 AM to minimise interference with other Internet activities.
- Data capture and analysis is suspended during log transfer.

All data capture is unintrusive to the trial participants and their usage of the Internet.

A. HTTP Analysis

The World Wide Web (WWW) runs over the HTTP Protocol [6]. Netsniff analyses all TCP traffic on port 80, 8000 or 8080 as HTTP traffic. The data collected for each HTTP transaction is:

- Start time and estimated duration of HTTP data transfer.
- Anonymised source and destination <*IP address; port number*> pairs.
- Estimated hop count between server and client.
- Running RTT and RTT jitter estimates sampled every ten seconds (utilising the algorithm presented by But et al [4]).
- Average RTT and RTT jitter for the duration of each flow.

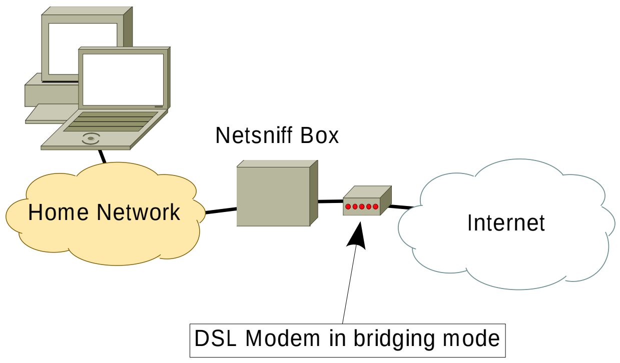

Figure 2: Netsniff deployment

- Estimated Packet Loss Rate in each direction of the dataflow, calculated using the Benko-Veres algorithm [13].
- HTTP Request type.
- HTTP Response from the server – includes HTTP Response code, content type, content cache-ability and content size (if present).

B. DNS Analysis

A HTTP request may be preceded by a DNS lookup to determine the IP address of the requested URL. Netsniff also captures all DNS traffic, logging:

- Time stamps of all DNS Packets
- Anonymised source and destination <*ip address; port number*> pairs.
- DNS ID numbers to allow matching between client request and server response packets.
- DNS response code.
- Anonymised IP addresses and URL's contained within the DNS packets. These IP addresses can be matched to the anonymised IP addresses in a HTTP transaction.

Captured DNS requests by a client can be matched to corresponding DNS responses by matching up <*IP address; port number; DNS ID*> triplets in the logged data. Once these packets have been matched, a DNS response time is calculated as the difference in the timestamps of the two captured packets. We acknowledge that this calculation does not take into account the RTT between the FreeBSD gateway and the client workstation, but given the experimental setup (Figure 2) where the two devices under consideration are in such close proximity, we estimate this error in DNS response time to be no greater than half a millisecond.

C. DNS and HTTP Correlation

Netsniff captures and analyses its data passively, it cannot directly determine which DNS query preceded a particular HTTP transaction. We can however extract this information from the logged data using the algorithm in (1)-(7), written in Extended Relational Algebra (σ: selection, π: projection, δ: Duplicate elimination, γ: aggregation, $\bowtie$: join). (1)-(3) define three tables with the captured DNS queries, DNS responses and HTTP connections respectively. (4) matches DNS queries and DNS responses by the ID field specified in RFC1035 [12], section 4.1.1. (5) joins HTTP and the combined DNS tables. (6) discards tuples, where the timestamp of the HTTP connection is smaller than the DNS response timestamp. (7) aggregates on qtime, rtime and selects the minimal HTTP timestamp.

$$DNS_Q = \pi_{qtime,id}(\delta(...)) \quad (1)$$
$$DNS_R = \pi_{rtime,id,ip\_res}(\delta(...)) \quad (2)$$
$$H = \pi_{htime,dst\_ip}(\delta(...)) \quad (3)$$
$$DNS_{QR} = DNS_Q \bowtie_{DNS_Q.id=DNS_R.id} DNS_R \quad (4)$$
$$H_{DNS0} = DNS_{QR} \bowtie_{DNS_{QR}.ip\_res=H.dst\_ip} H \quad (5)$$
$$H_{DNS1} = \sigma_{htime \geq rtime}(H_{DNS0}) \quad (6)$$
$$H_{DNS} = \gamma_{(qtime,rtime),min(htime)}(H_{DNS1}) \quad (7)$$

The resultant set contains all HTTP transactions where there is a preceding DNS query/response. There is the potential for error to occur in these calculations: Given that logs are rolled over daily, an edge case can occur where a portion of the DNS Query-DNS Response-HTTP Transaction occurs across the rollover period. In this case the matching will not occur.

We note that the potential occurrence of these unique cases is rare. The scheduled time for log rollover ensures that it is unlikely that any WWW traffic will be present at the time.

III. Experimental Results

A. DNS Resolution Time

First we analyse DNS lookup times without regard to correlating application information. The results are presented in Figures 3, 4 and 5. Figure 3 shows a cumulative distribution of response times for all captured DNS traffic. A total of about 123,500 DNS Query/Response pairs have been logged between the dates of September 2004 and January 2005. The graph provides some interesting results:

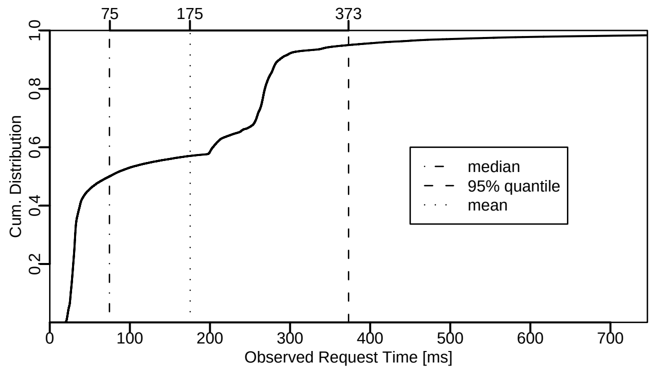

Figure 3: DNS lookup times for all traffic

- A large proportion (42%) of DNS requests are serviced within 40ms, representing resolution requests handled by the local ISP's DNS server[1] (either because the domain was local, or the result had been previously cached locally).
- A smaller portion (17%) of requests have a response time varying from 40-200ms.
- A number of requests (8%)have a response time varying from 200-250ms.
- The remaining 33% of requests have a response time greater than 250ms. These requests are also for entries that are not cached at the ISP DNS Server, however in this case it is necessary to obtain the information from an International DNS server. International links from Australia typically have a transport time of approximately 160ms[2]. The RTT within Australia and in other countries must then be added to this time.

Figure 4 is a cumulative graph of all DNS response times where there is a subsequent HTTP request following the DNS response, a total of about 73,500 requests. In other words, we restrict ourselves to DNS queries that form a critical part of WWW content retrieval time. In this case we see a greater proportion of DNS requests being serviced by the ISP's DNS server.

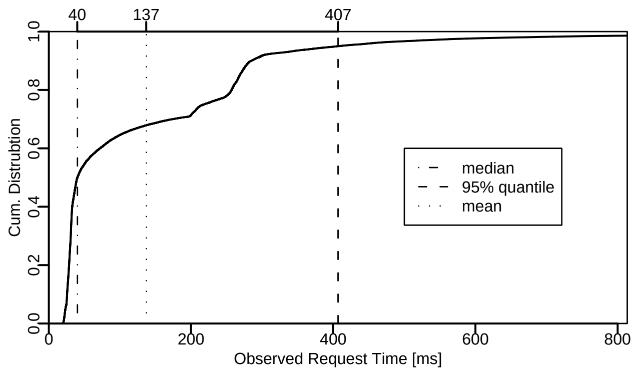

Figure 4: DNS lookup times for HTTP connections having a lookup

Not all HTTP transactions incur a previous DNS request. If the required IP address is already known to the browser being used, it can forego the DNS resolution request. Also, depending on OS implementation, DNS lookup results can be cached by the OS name resolver service. In either of these situations, no network traffic is generated to request a DNS resolution and the DNS lookup response time can be effectively considered to be instantaneous.

[1]Experimental lookup tests from each gateway to the corresponding ISP DNS server to a previously cached DNS name returned a lookup time of 40ms. We assume that this is due to the error-correcting interleaving mode used by our DSL providers.

[2]A traceroute to any IP address situated in the US typically reports a RTT from the last router in Australia to the first router in the US of about 160ms

Our traffic capture indicates that there were about 179,000 HTTP transactions in total, about 40% of which were preceded by a network based DNS request for an IP address.

Krishnamurthy et al. [9] suggest possible performance gains through piggy-backing HTTP requests onto the preceding DNS requests using the argument ”... Most HTTP transactions are preceded by a DNS lookup of the Web site’s server name ...”. Our data suggests that the potential gains would be minimised as on average only 40% of HTTP requests have a preceding DNS query and even less have a not locally cached preceding DNS query.

Figure 5 plots the DNS resolution times for all HTTP transactions, estimating a 0ms lookup time for situations where we estimate that the DNS resolution was cached either by the OS name resolver or the client browser.

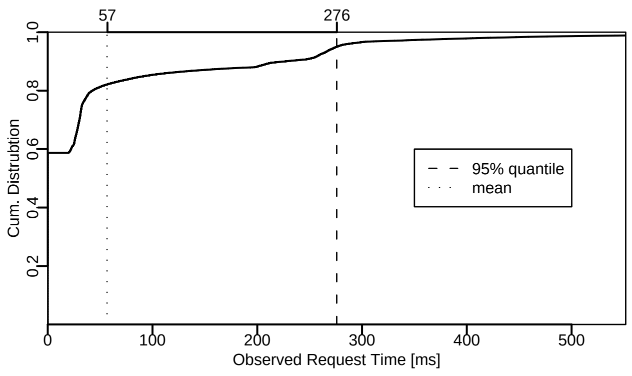

Figure 5: DNS lookup times for HTTP connections including browser cached entries

The implications of these results show that while DNS lookup time can in general be considered short, there are instances where URL resolution can add a significant time delay to the subsequent HTTP transaction. In at least 5% of all HTTP transactions, the overall transaction period is delayed by greater than 276ms while the name resolution is performed.

B. HTTP Transaction Time

We next look exclusively at data collated for network properties of logged HTTP transactions. Figure 6 plots the cumulative distribution of the estimated hop count between the client and the WWW server for all data. The graph clearly shows the lack of a WWW cache/proxy at the ISP, which would be indicated by a sizeable proportion of HTTP transactions with an estimated hop count of either one or two hops. This implies that all of our collected data has been for HTTP transactions directly to the source server.

If we consider the estimated Round Trip Times (RTT) of all HTTP transactions (Figure 7), the data indicates a reasonably linear spread ranging from 60ms (the estimated RTT between the client and the ISP network) and 450ms.

The linear spread is interrupted at about 120ms – a typical maximum RTT for destinations within Australia – resuming at about 220ms – a typical minimum RTT for destinations outside of Australia. The mean witnessed RTT is 351ms.

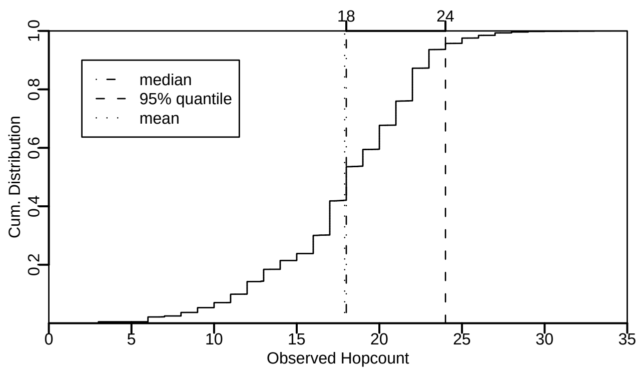

Figure 6: Distribution of path hop count of HTTP connections

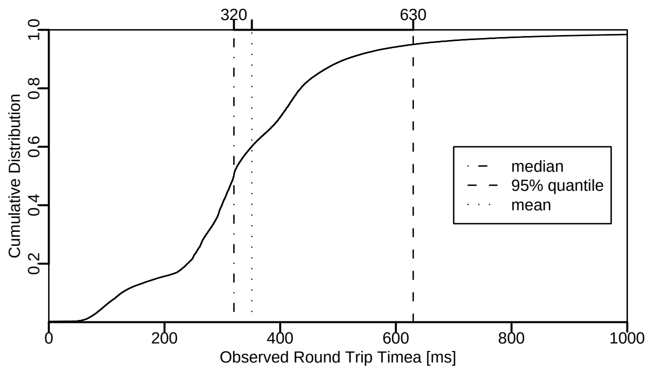

Figure 7: Distribution of RTT values of HTTP connections

Looking at the estimated network jitter (Figure 8), the data shows that nearly 80% of all HTTP transactions experience low jitter, while 95% of all transactions experience less than 125ms of jitter. The low value of jitter for a large number of streams is partially biased by the number of HTTP transactions of less than 4kB of data transfer – with less data packets to work with the estimated jitter is more likely to be low [4].

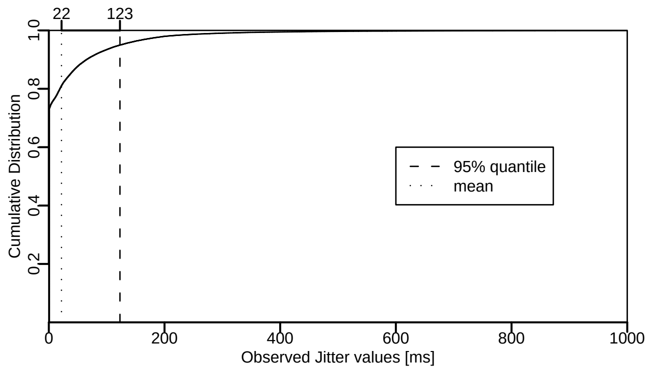

Figure 8: Distribution of Jitter values of HTTP connections

Figure 9 shows histograms of estimated RTT and estimated jitter for HTTP transactions with different hop count estimations between the client and server. This graph shows that both RTT and jitter are dependent on the number of network hops between the two communicating parties. We removed the value for hop count 5 from Figure 9 (the value resulted from connections to a single destination with apparent network problems).

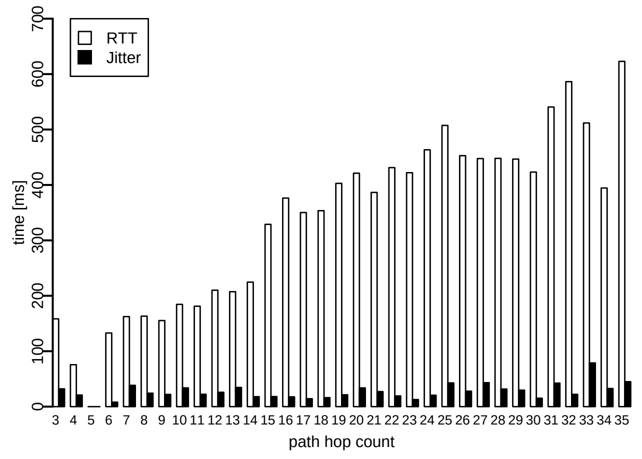

Figure 9: RTT/Jitter versus path hop count for HTTP connections

Figure 10 plots both the estimated RTT and estimated jitter in order of increasing RTT. The graph indicates that network jitter is correlated to RTT, and decreasing the RTT by decreasing the hop count to the server can have the flow-on effect of decreasing the network jitter and improving network throughput. G. Armitage and L.Stewart present a more extensive discussion in [2].

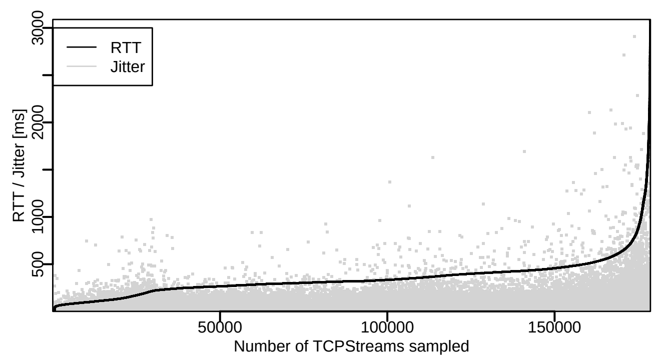

Figure 10: Dependency of Jitter on RTT for HTTP connections

The implications of these results show that local caching of content has the potential to improve network conditions for data transfer. This would provide a constant, low, hop count to the cache with subsequent low and predictable RTTs and network jitter parameters.

C. Cacheability of Content

The netsniff application determines content cacheability via examination of the HTTP server response following a GET request. The determination is made using rules described in section 13.4 of RFC 2616 [6]. This allows determination of whether content transferred as part of an HTTP transaction is cacheable or not.

A second class of cacheable content exists, this is where the content is currently cached by the client browser and the browser requests an update of the cached content if necessary. If the cache entry does not require updating, the server responds with an HTTP Response code 203 – indicating that the existing copy of the content is current.

In this case we have a scenario where no content is actually transferred and the simple cacheability check returns no detail on the cacheability of the content. However, the content was obviously cacheable – as it is already had previously been cached by the browser.

Figure 11 shows the proportion of non-cacheable and cacheable objects separated by content type. Previously cached objects – as detected via HTTP Response Code 203 – are displayed under the label *"previously cached"*, the actual content type cannot be determined via passive measurements. The displayed percentages are for the total number of HTTP transactions. Figure 12 repeats the same

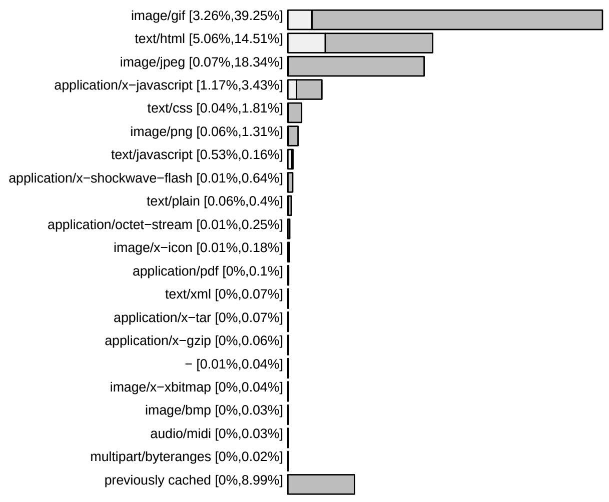

Figure 11: Cacheability of content types by count

data which has been recalculated for percentage of total size of the downloaded content.

These results, and the cumulative distributions of cacheable and non-cacheable web object sizes (see Figure 13 and 14) mirror those determined by Zander et al. [15]. Our measurements, made in a real, home Internet access environment validate the previous results produced via an automated web access procedure.

D. Overall HTTP transaction performance

In this section we consider the time required for a *"typical"* HTTP transaction. Figure 1 shows the time periods involved in a HTTP transaction:

- **Initial DNS lookup time** – If the required data has been cached either by the browser or the clients OS

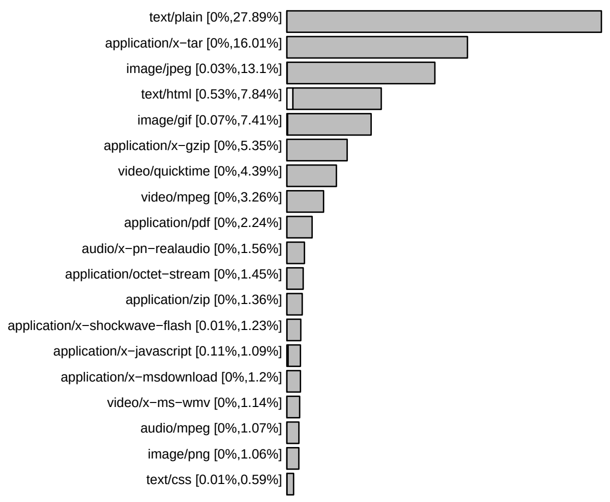

Figure 12: Cacheability of content types by size

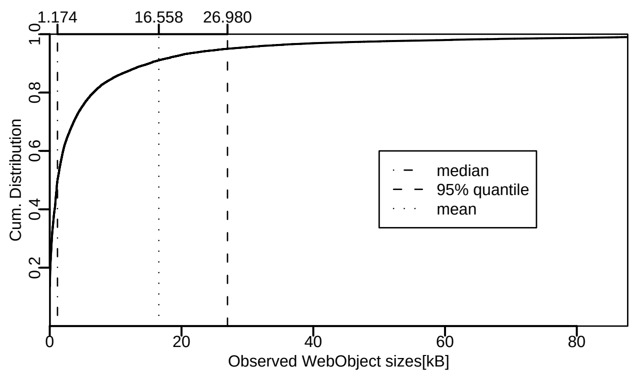

Figure 13: Distribution of cacheable Web object sizes

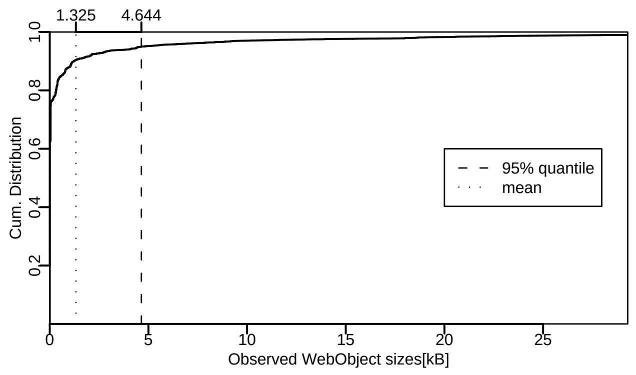

Figure 14: Distribution of not cacheable Web object sizes

name resolver, then this time period is effectively instantaneous.

- **Browser think time** – The time taken by the browser to initiate the TCP connection to the WWW server once it has the required IP address. This time is estimated as the time between receiving a DNS response and initiation of the first SYN packet from the browser.
- **Data Wait Time 1** – A TCP connection is initiated via a SYN-SYN/ACK-ACK handshake sequence between the two communicating parties. In the case of no packet loss, the first two packets of this sequence take a time period equal to the RTT between the two communicating parties.
- **Data Wait Time 2** – The last packet of the TCP handshake (ACK) is either:

  1. Immediately followed by a data packet containing the HTTP request.
  2. Has the HTTP request appended on to the last ACK packet of the TCP handshake.

  The server will then respond with the first packet of data containing the response. Again in the case of no packet loss, this time is approximately equal to the RTT between the two communicating parties.
- **Data Transfer Time** – The time taken to transfer the HTTP response to the client. Given that netsniff cannot accurately determine this time due to browsers holding HTTP connections open longer than the actual data transfer, we estimate this time as the required time to transfer the content given its size at the client ADSL line rate of 512kbps. While this time could be longer – particularly for servers at a greater distance to the client – the calculated time does place a lower bound on the time required for content transfer.

Figure 15 shows the mean HTTP transaction time for transactions involving both cacheable and non-cacheable objects, calculated using the mean time periods observed for each portion of the transaction. The mean DNS lookup time has a lookup time of 0ms factored in for observed cases where the DNS result was cached at the client workstation.

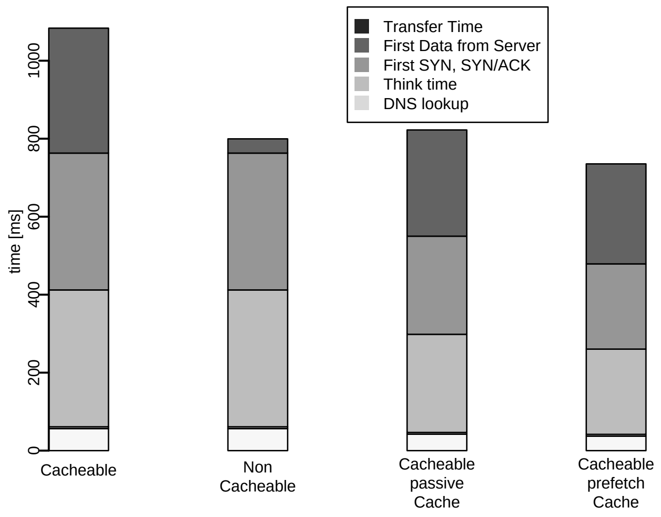

Figure 15: Characteristics of average HTTP connections

What is particularly obvious is that a large portion of the transaction time is due to the TCP handshake between the browser and the WWW server. The potential benefits of reducing this time by using a local cache with a much lower RTT is obvious.

Let us consider the implications of using a local cache:

- If a proxy cache is set by the browser, then the IP address of the cache will be locally cached and no DNS resolution will need to be performed. This will reduce the initial DNS lookup time to 0ms. However if transparent proxying is enabled then the client browser will still perform a DNS lookup for the requested page.
- Using the network under measurement, a browser communicating with a cache sited at the ISP will experience a RTT of about 20ms with practically no network jitter. This will not only increase throughput, but also reduce the time required for the TCP handshake to occur.
- Content transfer will be quicker and more likely to approach our "calculated" value based on ADSL line speed. Given that available ADSL bandwidth limitations are partially artificial based on the plan the consumer pays for, this time can be further reduced by the ISP configuring the network to not bandwidth limit traffic coming from the ISP internal network and cache.

However, not all cacheable content will be cached. Whether content will exist on the cache is dependent on the hit rate experienced by that cache, where hit rate is determined by a number of factors, including the size of the cache and the number of users utilising the cache. In the case where a request for content is made that results in a cache-miss, the following penalties are applied:

- If the IP address of the WWW server is not known, the cache must still perform the DNS lookup.
- The page must be obtained from the WWW server, taking the same amount of time as in the case with no cache.
- There is an additional penalty based on the think-time of the cache as it determines whether or not it can satisfy the request without consulting the original server.

We estimate the potential performance improvements on our home consumer by modifying the DNS lookup time by assuming that for the case where a cache hit occurs the DNS lookup time is 0ms. We also lower the mean RTT such that we consider a cache hit to incur an RTT of 20ms.

These modifications are performed using an estimated hit ratio of 30%, where we assume a purely passive cache such as Squid [1]. Results from the NLANR's IRCache project [14] have shown the cache hit ratio can reach up to 30% for a well populated cache. More recent approaches use predictive prefetching to reduce latency [7, 5] can achieve even higher hit rates, we assumed a 40%, however note that predictive prefetching trades off bandwidth for perceived latency.

The effects on overall transaction time are plotted in Figure 15. The results indicate a possible improvement in content access time of the order of 250-350ms (24-32%).

It is interesting to compare these cache hit ratios against the proportion of cacheable content that is accessed. Given that around 88% of accessed content is cacheable (see Figure 11 and [15]), there is the potential for a much higher hit ratio to be seen, perhaps through an increase in both the storage capacity and the number of serviced users of a cache.

While these show an immediate potential benefit to the consumer today, they provide other less tangible benefits to all Internet users. A decrease in network traffic due to requests being service at the edge of the network will lead to a decrease in network congestion and a subsequent performance increase in non-cacheable peer-to-peer applications thanks to a higher network throughput and a lower packet loss rate.

IV. Implications for Other Online Applications

As previously mentioned, the WWW is only one of many Internet applications. What lessons can we learn from the current widespread usage of the Web that can then be applied to other online entertainment applications. We have already mentioned the potential benefits on other Internet applications with the increased use of WWW caching. Our data gathering environment was set up using two different ISPs, both specifically not offering web caching services – a statement verified from our empirical evidence.

The decision not to offer caching of web content could be based on the ill-founded assumption that a higher bandwidth connection of 256/512/1500kbps does not require local caching of content to improve network performance. It is true that the perceived performance is better using current broadband access technologies when compared to dial-up modem access, however this is primarily due to the decreased content transfer time which is possible with a higher capacity link. The results in this paper imply that caching of web content by an ISP would improve the perceived performance of their Internet offerings through decreased average page load times.

However, this perceived performance improvement comes at a cost. There is the possibility that cached content may be stale, dependent of the time required for updated content to propagate to caches throughout the Internet.

The WWW is not the only online application that would benefit from caching. Newer online application such as streaming audio and video would greatly benefit from a caching approach. Indeed, any application whereby the distributed content either does not change or does not change regularly, can benefit through the use of cached content.

Let us consider a streaming video application. In this case the issues of TCP handshaking and RTT is not relevant. Caching reduces the jitter and hop count of the network path taken to stream content, however this is a problem that can also be addressed via increased buffering at the playback client [11]. More important is the potential savings of bandwidth usage in the core of the network through the use of caching [3]. This provides a scenario whereby not only is scalability improved, but also where network resources are freed up to support other applications.

As for web traffic, there are also downsides to caching video and audio content, particularly if this content is nonstatic and may occasionally change. The propagation of updated content through the Internet and onto caches may lead to users fetching stale content. Also, when considering entertainment based applications, there is a greater likelihood for much of the content to be commercial in nature – introducing the aspect of copyright protection of content distributed onto various caches [10]

True peer-to-peer applications such as online multiplayer gaming, Internet phone calls and live streaming of video cannot utilise caching directly. However these applications benefit from increased caching of content by applications that can, greater available network resources in turn improve the perceived performance of peer-to-peer applications.

V. Conclusion

Usage of the Internet is growing at a phenomenal rate – the Internet is being used as a form of entertainment rather than just an information tool. It is expected not only that this growth will continue, but also that existing applications will expand in functionality and that new entertainment options will be developed.

In this paper we have discussed the perceived performance of the WWW application as captured at ”real” residential Internet sites. Our results demonstrate where the performance issues lie with respect to the WWW. From these results it is clear that there are many potential benefits to be had by increased caching of content by the end users ISP, with a perceived improvement in content access times of the order of 24-32%.

Caching does not only benefit the WWW application, the principles of content caching can also be applied to other applications where content remains relatively static (e.g. streaming video). Content caching can also benefit other online applications such as online multiplayer gaming. These peer-to-peer applications can not make direct use of caching techniques but instead can benefit from increased network resources that would be made available due to caching in applications where this is possible.

References

[1] Squid Web Proxy Cache, 2005. http://www.squid-cache.org/.

[2] G. Armitage and L. Stewart. Limitations of using Real-World, Public Servers to Estimate Jitter Tolerance Of First Person Shooter Games. *ACM SIGCHI ACE2004 conference*, June 2004.

[3] J. But and G. Egan. Designing a Scalable Video-On-Demand System. *International Conference on Communications, Circuits and Systems (ICCCAS02), Chengdu, China*, pages 559–565, June 2002.

[4] J. But, U. Keller, D. Kennedy, and G. Armitage. Passive TCP Stream Estimation of RTT and Jitter Parameters. *Submitted to ACM Computer Communications Review*, January 2005. http://caia.swin.edu.au/cv/jbut/publications.html.

[5] B. C., K. A., and K. D. Efficient Reduction of Web Latency through Predictive Prefetching on a WAN. *Lecture Notes in Computer Science, Springer-Verlag GmbH*, 2762:25–36, September 2003.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999. http://www.ietf.org/rfc/rfc2616.txt.

[7] T. I. Ibrahim and C.-Z. Xu. Neural nets based predictive prefetching to tolerate WWW latency. In *International Conference on Distributed Computing Systems*, pages 636–643, 2000.

[8] U. Keller and J. But. Netsniff - Design and Implementation Concepts. Technical Report 050204A, CAIA, February 2005. http://caia.swin.edu.au/reports/050204A/CAIA-TR-050204A.pdf.

[9] B. Krishnamurthy, R. Liston, and M. Rabinovich. Dew: Dns-enhanced web for faster content delivery. *In Proceedings of the Twelfth International World Wide Web Conference, Budapest, Hungary*, May 2003.

[10] J. Lee, S. Hwang, S.-W. Jeong, K. Yoon, C. Park, and J.-C. Ryou. A DRM Framework for Distributing Digital Contents through the Internet. *ETRI Journal*, 25-6:423–436, December 2003.

[11] C.-W. Lin, J. Zhou, J. Youn, and M.-T. Sun. MPEG Video Streaming with VCR Functionality. *IEEE Transactions on Circuits and Systems for Video Technology*, 11-3:415–425, 2001.

[12] P. Mockapetris. Domain names - implementation and specification. RFC 1035, IETF, Nov. 1987. http://www.ietf.org/rfc/rfc1035.txt.

[13] P. Benko and A. Veres. A Passive Method for Estimating End-to-End TCP Packet Loss. *Proceedings of IEEE Globecom, 2002*, November 2002. http://www.comet.columbia.edu/~veres/globecom02.pdf.

[14] Wessels D. and Rousskov A. NLANR cache hierarchy statistics,visualization, and roi, February 1998. http://www.caida.org/outreach/presentations/Nanog9802.

[15] Zander S. and Armitage G. Dynamics and Cachability of Web Sites: Implications for Inverted Capacity Networks. *11th IEEE International Conference on Networks (ICON 2033), Sydney, Australia*, September 2003.