Scalable Web Caching of Frequently Updated Objects using Reliable Multicast

Dan Li and David R. Cheriton
Stanford University

Abstract

Frequently updated web objects reduce the benefit of caching, increase the problem of cache inconsistency, and aggravate the inefficiency of the conventional "repeated unicast" delivery model. In this paper, we investigate multicast invalidation and delivery of popular, frequently updated objects to web cache proxies. Our protocol, MMO, groups objects into volumes, each of which maps to one IP multicast group. We show that, by forming volumes of the appropriate size and/or object correlation, the benefit from reliable multicast outweighs the cost of delivering extraneous data as well as the overhead of multicast reliability. Moreover, trace-driven simulations show that the bandwidth saving over conventional approaches increases significantly as the audience size grows. We conclude that MMO provides efficient bandwidth utilization and service scalability, and makes strong web cache consistency for dynamic objects practical.

1. Introduction

Web proxy caching [4] is critical to the continuing success of the Web. It improves the response time and reduces the load on the network and web servers. The falling cost of memory and disk allows web cache proxies to hold an increasing amount of web content. As the Web carries more web objects[1] that are *both* accessed and modified frequently, the hit rate of web caches is limited more by consistency than by cache capacity. Cached copies of frequently updated objects become *stale* more often. Frequently retrieving new copies defeats the benefit of caching.

Frequently updated objects also raise the consistency protocol overhead. With web cache consistency protocols such as adaptive TTL (Time-To-Live) [15], the rate of polling by the proxy must be considerably higher than the rate of modification at the web server in order to maintain an acceptable *stale rate* (percentage of instances that the cache returns a stale document). For frequently updated objects such as sports and financial news that change several times a day, polling overhead can be excessive for the network and the web server. Alternatively, the web server can send cache invalidations to web caches. Cao et. al. [7] performed an excellent study on a TCP-based invalidation protocol, concluding that strong cache consistency can be maintained with little or no extra cost over the current weak-consistency approaches. However, the web server has to keep per-proxy state and establish TCP connections to all of the proxies to deliver the invalidation, a significant overhead for widely cached objects. Moreover, after the invalidation, there is likely to be a sudden influx of requests from many caches (triggered by client requests or prefetching [24]), potentially saturating the server and causing link congestion. These bursts of requests may produce peak loads comparable to that experienced without caching. If servers are engineered for these peak loads, the benefit of caching for servers is minimal. Fundamentally, the "repeated unicast" delivery model does not scale.

Addressing these problems, we propose MMO — multicast invalidation followed by multicast delivery of a *volume* of web objects to web cache proxies using the OTERS reliable multicast protocol [20]. The cost of OTERS in this context is evaluated using NS [30]. We study MMO's performance using trace-driven simulations. From these studies, we conclude that MMO is far more scalable than conventional and hybrid protocols and provides strong cache consistency, fast responses, and efficient bandwidth utilization.

The rest of the paper is organized as follows. Section 2 describes our proposed protocol. Section 3 discusses a number of alternatives to be compared with our approach. Section 4 outlines the simulation environment. Section 5 assesses the cost of unicast and multicast transport protocols. Section 6 analyzes the performance of our protocol. Section 7 discusses related work. Section 8 concludes the paper. Appendix A describes web access traces. Appendix B describes the process of measuring the transport protocols.

2. A Multicast-based Web Caching Protocol

In MMO, the web server multicasts cache invalidations

1 A web object consists of one or more files a browser needs to retrieve from the web server in order to display a URL. A web server (or server) refers to a web content source; a web cache proxy (also as a cache or a proxy) refers to a shared URL cache for a group of local web clients, e.g., hosts within an ISP network or a corporate LAN.

and modified objects to a multicast group using the OTERS reliable multicast protocol. Web caches subscribe to the multicast group to receive the information.

2.1. OTERS

OTERS (On-Tree Efficient Recovery using Subcast) [20] organizes group members into hierarchical subgroups by exchanging session messages to elect a *designated receiver* or *DR* for each subgroup. DRs then employ *subcasting*[2] for local retransmission. Figure 2.1 shows its recovery process.

The notification mode (OTERS-NT) uses ACKs to reliably deliver notifications such as web cache invalidations. Upon receiving a notification, the group member sends an ACK to its DR, which in turn sends an ACK to its own DR. Each DR retransmits the notification to non-responding subgroup members.

The file transfer mode (OTERS-FT) is designed for files. A receiver learns about parameters of the file transfer from a prior notification (e.g., a web cache invalidation), including the starting time, the file length and the transmission rate. At the end of the file transmission, if the receiver has missed any packets, it sends a NAK (containing the sequence numbers of all missing packets) to its DR for retransmissions.

2.2. The Invalidation Phase

In any invalidation protocol, the web server sends invalidation messages to web caches when an object is modified. Each web cache then deletes the cached copy (if one is cached). In the presence of network or process failures, *leases* or *volume leases* [12, 33] can serve as an efficient fault-tolerance mechanism. Currently the HTTP protocol does not support invalidation but the server part can be implemented in the HTTP accelerator [8] (a form of web caching at the server location). This way, invalidation becomes part of a signaling protocol between web caches such as ICP [32].

In MMO, an *invalidation channel* is a multicast address that web cache proxies subscribe to. The web server uses OTERS-NT to notify the channel subscribers of modifications to a volume of objects. A proxy that is not subscribed to the invalidation channel still requests those objects directly from the web server, which indicates in the response that an invalidation channel exists.

[2] Subcasting is multicasting of a packet over a subtree of the multicast delivery tree. One subcast retransmission can repair an entire subtree's losses that are caused by one packet drop at the root of the subtree. OTERS is built on IP encapsulation [26] and IGMP traceroute [10], with security extensions that involve router changes but impose no additional state and little processing overhead.

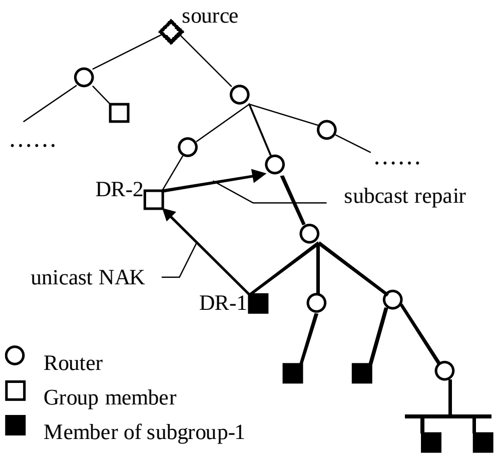

Figure 2.1 OTERS. *Designated Receiver* DR-1 unicasts a NAK (negative acknowledgment) to its own designated receiver − DR-2 − after detecting a packet loss. DR-2 responds with a repair to DR-1's subtree, assuming DR-2 received this packet. Bold links indicate the path of the retransmission.

The proxy can join the channel when enough number of objects in the volume is cached.

The invalidation channel is expected to be long-lived and have relatively stable memberships. For example, a proxy may stay in the channel for 12 hours or longer. Using OTERS also means that subscribed proxies are committed to exchanging information and maintaining the subgroup hierarchy. Fortunately, web caches are generally stable, well maintained, and well connected to the Internet. The opposite is dial-up or wireless users, for whom our scheme is not suitable.

2.3. The Delivery Phase

After a multicast invalidation, the server multicasts the modified object via OTERS-FT to the invalidation channel, which subscribed proxies receive and then return to clients in subsequent requests.

A *volume* is a set of objects that share the same invalidation channel. Having multiple objects per volume is more efficient than having one object per volume because the *multicast overhead* (including address allocation, routing and transport session organization) can be amortized over more objects.

A multi-object volume introduces extraneous data. For instance, one may receive from the multicast channel a message that invalidates an object it does not cache, or receive an object that is modified again before any client requests the object. The server, however, can reduce the amount of extraneous data by limiting the volume size and assigning related objects to the same volume (so that a proxy is likely to cache most of them). The server can form volumes based on access statistics,

Acronym	*Invalidation Method*	*Delivery Method*
MMO	Multicast via OTERS-NT	proactive Multicast via OTERS-FT
MMF	Multicast via OTERS-NT	proactive Multicast via Digital Fountain
UMF	Unicast via TCP	proactive Multicast via Digital Fountain
MU	Multicast via OTERS-NT	on-demand Unicast via TCP
UU	Unicast via TCP	on-demand Unicast via TCP
AT	Adaptive TTL	on-demand Unicast via TCP
PET	Polling-Every-Time	on-demand Unicast via TCP

Table 1 Acronyms of the seven web caching protocols

URL prefixes, content subjects, etc. [9]. Furthermore, proxies are less prone to extraneous data than end-users because a proxy aggregates requests from many end-users, raising the traffic and hit rate of popular objects.

2.4. The Pros and Cons of MMO

MMO offers several significant advantages. First, multicast invalidation provides strong cache consistency[3] without any per-proxy state at the server and without aggressive polling by the proxy. Second, proactive multicast updates provide lower web access time than on-demand unicast delivery. Third, multicast invalidation and delivery are more scalable to large audiences than their unicast counterparts.

However, proactive multicast is not always more efficient than on-demand unicast because of multicast overheads and extraneous data. But MMO compensates for these potential drawbacks by employing one channel for both cache invalidation and object delivery to amortize multicast overheads. MMO also relies on efficient volumes to control the amount of extraneous data. Hence MMO is more efficient when delivering popular, frequently modified and correlated web objects in a volume to a large number of web caches. For example, the CNNfn.com homepage and top stories can be disseminated in a volume using MMO.

3. Other Web Caching Protocols

To set the stage for comparing MMO with other alternatives, we first introduce some hybrid protocols (MMF, MU, UMF and UU) along with the traditional ones (AT and PET). Table 1 lists their main features.

3.1. Hybrid Web Caching Protocols

MMF and MU use multicast invalidation, similar to MMO. Conversely, UMF and UU use *unicast invalidation* [7]. The web server keeps a list of web caches that have requested an object since its last modification. When the object is modified, the server sends an invalidation (via TCP) to each cache on the list.

MU and UU use *on-demand TCP delivery*. After the invalidation (either unicast or multicast), the proxy deletes the invalidated copy and retrieves a fresh copy only when the next client request arrives. There is no extraneous data but the server has to repeatedly unicast the object on demand.

MMF and UMF use *proactive multicast delivery via Digital Fountain*[4]. After the invalidation (either unicast or multicast), the server multicasts the modified object via Digital Fountain to a *delivery channel*, a multicast group that is allocated for delivering this object). The delivery channel is short-lived. Proxies can decide whether to join it. If a proxy joins the channel, it receives a copy and returns this copy to clients upon future requests. Otherwise, it retrieves a fresh copy (via TCP) when the next client request arrives.

The delivery channel allows a tradeoff between unicast and multicast delivery methods, in the amount of extraneous data a proxy chooses to receive. The prefetch decision is based on the probability of a future client request coming before the next modification. We use the following policy. Define a *join threshold W*. If no client has requested the object for the time spanning the last *W* invalidations, the proxy does not join the delivery channel. Otherwise, it does. With this flexibility, MMF incurs the multicast overhead on each delivery. Section 6 shows that MMO in fact outperforms MMF.

To efficiently support the above schemes, there are two requirements on multicast routing. First, the invalidation channel is long-lived and requires efficient routing state maintenance, e.g., limiting membership heartbeats to occur only at the leaves. Second, the delivery channel is short-lived and requires fast join/leave and scal-

[3] There is a small window of opportunity (from the creation of cache invalidation to the completion of object delivery) for clients to get the slightly obsolete copy from the proxy.

[4] Digital Fountain [5, 29] is designed for bulk data transfer. The source encodes an entire file using Forward Error Correction codes and multicasts it continuously by looping through the encoded data. A receiver tunes to the multicast channel at any time and leaves the channel as soon as it receives enough encoded packets in order to reconstruct the original file. The source can stop sending once the multicast group is empty or after having looped several times.

able address allocation. These requirements are consistent with the research community's effort on multicast routing and are met by proposals such as EXPRESS single-source multicast routing [17].

3.2. Traditional Web Caching Protocols

Polling-every-time provides strong cache consistency, like all the above protocols. The proxy always sends an "If-Modified-Since" request to the server before returning any cached copy to clients. The server responds with either a modified copy or "Not Modified". The latter case is called a *slow hit* because the cached copy is returned to the client after a round trip to the server. Conversely, in an invalidation-based protocol, all hits are *fast hits* because the cached copy is immediately returned to the client.

Adaptive TTL [15] provides weak cache consistency and is based on the observation that "older" files are less likely to be modified. The proxy sets the TTL of a cached copy to α times the "age" of the object (i.e., from its last modification to now). By default, α is 0.2 in Squid [32] and 0.5 in Harvest [4]. Before the TTL expires, client requests are served directly from the cache. They are fast hits but may be stale. Upon the first client request after the TTL expires, the proxy sends an "If-Modified-Since" request to the server. The result may be a modified copy or a slow hit. Then the TTL of the cached copy is adjusted accordingly.

4. The Simulation Environment

4.1. Web Access Traces

The simulation uses three types of traces. One is the *Surge trace*, generated by the Surge HTTP request generator [2]. Second is the *Stanford trace*, the server log of Stanford University's official web site. Third is the *NLANR trace*, proxy logs of accesses to *CNN.com* by the 8 top-domain proxies in the NLANR (National lab of Applied Network Research) Cache Hierarchy [23]. *Appendix A* describes these traces in more detail.

Generate modifications. The traces do not provide the object modification history so we adopt the *hot/cold* model [7] to generate modifications. First, 1% of the web objects are picked uniformly across the object popularity ranks as the frequently updated (or hot) objects. Then, given *k* hot objects and an average object lifetime of *L* seconds, every *L / k* seconds the modification generator randomly picks one from the *k* objects to modify. This leads to a geometric lifetime distribution.

Volume formation. The Surge and Stanford traces use *random formation*. In other words, a volume of size V consists of the V most popular, frequently updated objects. The NLANR trace uses *prefix formation*. The volume consists of six objects that share the URL prefix "*http://www.CNN.com/WORLD/meast/9812/17/iraq.strike*".

4.2. Join Decision and Caching Decision

A proxy joins the invalidation channel if it caches at least one object in the volume. In reality, a proxy may decide to join the channel only after a few objects in the volume are cached. Our assumption is more conservative in that it results in more extraneous data.

After the proxy joins the channel, any object in the volume is cached once accessed. This decision is realistic because objects in the volume are popular (based on the server's statistics) and warrant caching. Caching objects in the volume that are less popular to a proxy presents only disk space cost and no extra consistency cost. With cheap disks and RAM, a deep cache or cache farm can afford the space in exchange for lower bandwidth consumption and better response time to the end-users. A volume-wise caching decision also does not reduce the hit rate in a deep cache because a cache often can reach a size beyond which the hit rate does not rise much by adding more cache space. For example, a 24GB cache is sufficient for a daily web flow of 100 gigabits (according to the ISP-caching mailing list).

4.3. Performance Metrics

The web-caching simulation uses the following performance metrics:

response time: the time from when the proxy receives a client request to the time it finishes responding. This metric reflects the user-perceived web access time because the way a client contacts its proxy is the same regardless of the web caching protocol.

stale rate: the percentage of responses a proxy returns to its clients that contain stale data. Only adaptive TTL has a non-zero stale rate. All other protocols offer strong consistency and therefore zero stale rates.

packet count: the number of distinct packets exchanged among the web server and proxies in order to fulfill the client requests. Packets that a proxy sends to its clients are not counted because all the protocols incur the same cost. The packet size is assumed to be 1024 bytes, a compromise between two popular network packet sizes: 550 and 1500 bytes.

packet-hop count: the number of hops the packets tra-

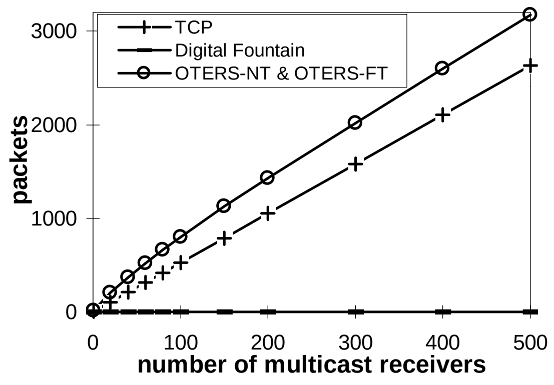

Figure 5.1 Session Overhead in packets

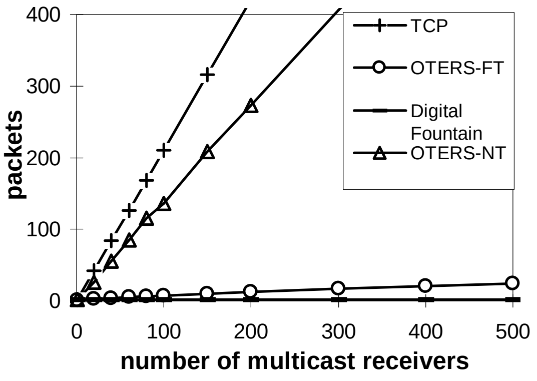

Figure 5.2 Per-Packet Cost in packets

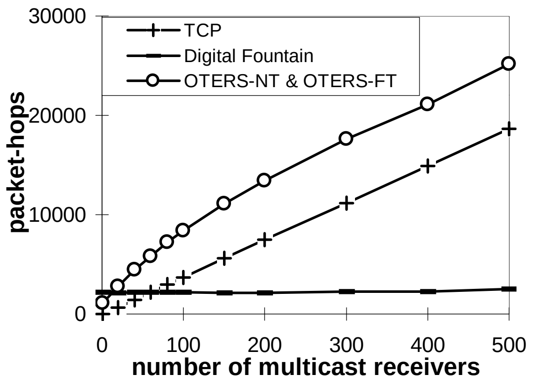

Figure 5.3 Session Overhead in packet-hops

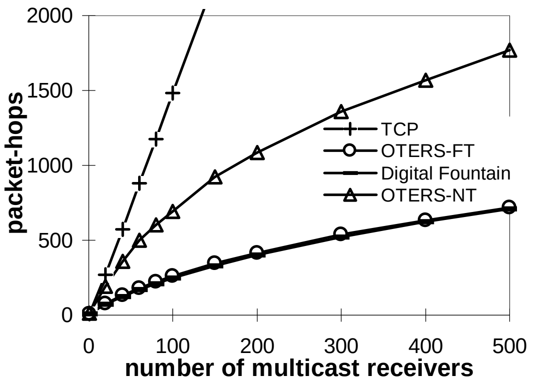

Figure 5.4 Per-Packet Cost in packet-hops

verse between the server and proxies, reflecting the amount of wide-area traffic a caching protocol imposes.

5. Performance of the Transport Protocols

To assess the traffic load of the various web-caching protocols, we developed a traffic load model for the transport protocols TCP, OTERS and Digital Fountain. *Appendix B* describes the measurement process on the simulator NS [30]. Measurements show that the traffic load of a transport session can be modeled as *Load(m,n) = f(m) + n • g(m)*, where *f(m)* is the session overhead, *g(m)* is the per-packet cost, *m* is the number of multicast receivers and *n* is the number of payload packets.

TCP's overhead and per-packet cost are linear in m. TCP's overhead comes from the 3-way handshake and the connection termination. The overhead of OTERS comes from organizing the subgroup hierarchy. The overhead of Digital Fountain comes from packets that the network delivers after a receiver has received all that are necessary to reconstruct the original file but before its leave message is propagated all the way up the multicast delivery tree. The higher rate the source transmits, or the slower the leave message propagates, the more Digital Fountain overhead. Additional overhead may come from flooding of the initial multicast packet, e.g., in a DVMRP routing domain [31].

Figures 5.1 and 5.2 plot the packet counts of the session overhead and the per-packet cost respectively. Figures 5.3 and 5.4 plot the packet-hop counts. The session overhead of Digital Fountain is significantly less than that of TCP and OTERS. However, in MMF, the Digital Fountain overhead is amortized over a single delivery, while in MMO the OTERS overhead is amortized across multiple deliveries. The per-packet costs of OTERS-FT and Digital fountain are similar and much lower than that of TCP and OTERS-NT because the former two use NAKs while the latter two use ACKs.

6. Web Caching Performance Analysis

The traces were replayed through a web caching simulator that implements the 7 protocols (see also Table 1). Performance data is gathered over requests to objects in a volume. Requests outside the volume were not considered. Every set of results has three parameters: *V* — the number of objects in the volume, *P* — the number of proxies, and *L* — the average object lifetime (in minutes). In adaptive TTL, α is set to 0.25. In MMF and UMF, the join threshold *W* is set to 1.

6.1. From the Client's Perspective

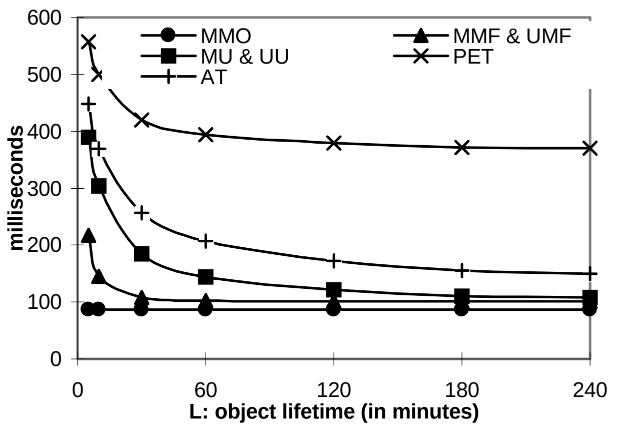

Figure 6.1 Average Response Time (NLANR, P=8, V=6)

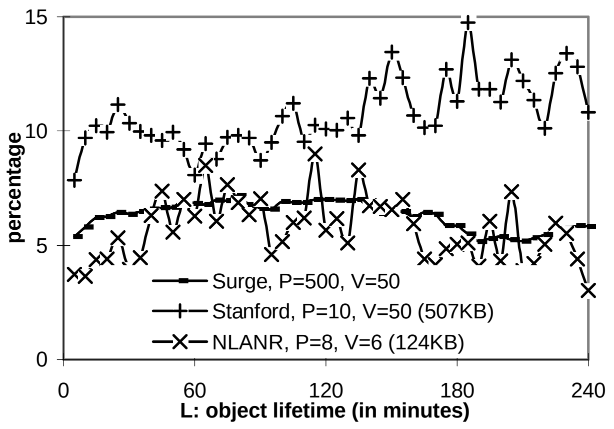

Figure 6.2 Stale Rate of Adaptive TTL

The response time and the stale rate quantify the service quality that end-users experience. Figure 6.1 plots the average response time relative to *L*, for the NLANR trace. MMO reduces the response time to 57% of that of AT when *L* = 4 hours and to 34% when *L* = 30 minutes. MMO sets the lower bound because it generates only fast hits. Other protocols' curves, however, shoot up as the object lifetime shortens, causing more of their fast hits become slow hits or misses. MMF is faster than MU because it retrieves objects sometimes proactively and sometimes on demand. AT polls the server once TTL expires and may discover the document is not modified. Therefore AT has a higher response time than all the invalidation-based protocols. PET is the slowest because it polls the server on every request.

Figure 6.2 plots the stale rate of AT for the three traces. It shows that, with $\alpha = 0.25$, AT can reach a stale rate of 5% to 15% for objects modified more than once every four hours. The Stanford trace has a higher stale rate because it directly records the end-users' access pattern and hence has more *clustered requests* (requests to the same object, e.g., a course's announcement page, that occur within a short interval, e.g., 3 hours). With clustering, more requests occur before the TTL of the cached copy expires and are subject to stale responses. Conversely, requests to NLANR top-domain caches are filtered by lower-level caches. Requests generated by Surge are also relatively spaced out.

6.2. From the Server's Perspective

Packet counts indicate the amount of traffic that servers and caches have to generate to deliver the web content. Figures 6.3 and 6.4 plot the packet count vs. *L* and *V* respectively for the Surge trace. Figures 6.5 and 6.6 plot the same for the Stanford trace. Figures 6.7 and 6.8 plot the packet count vs. *P* for Surge and NLANR respectively. The figures' Y axes vary in their ranges but all cover 3 magnitudes in logarithmic scale for easy relative comparison, except that Figures 6.7 and 6.8 use a linear scale to show the tangent of the curves.

Overall, MMO sets the lower bound and PET sets the upper bound. Figure 6.3 shows that, with 500 proxies, MMO is over an order of magnitude more efficient than MMF and UMF, and almost 2 orders of magnitudes more than AT and PET. MMF and UMF are close to each other. So are MU and UU, indicating that delivery (as opposed to invalidation) accounts for the majority of the traffic.

Figures 6.3 and 6.5 show that the traffic load increases as the object lifetime decreases. The increase is more significant for MMO than for unicast-based protocols like PET because, as the lifetime shortens, more cached copies are not referenced before being invalidated again. Volume size also affects the amount of extraneous data multicast delivered. On one hand, the web server would like to include as many objects as possible in one volume in order to amortize the multicast overhead. On the other hand, as the volume grows, the traffic load of MMO rises faster than that of unicast-based protocols (Figure 6.6). In this case, the Stanford web server should choose a volume size of 50 or less.

Despite the extraneous data, multicast-based protocols perform much better than their unicast counterparts when the number of proxies is large. Figures 6.5 and 6.6 have just 10 proxies. Figure 6.3 (500 proxies) shows that MMO outperforms other protocols even with 5-minute object lifetime. Similarly, in Figure 6.4 (500 proxies), MMO does not even reach the magnitude of AT's traffic load at volume size 100, meaning that it can carry up to 1000 objects in a volume and still outperform AT carrying 100 objects. This is because multicast scales to large audiences with little increase of traffic. For example, Figure 5.2 shows that OTERS-FT uses 24

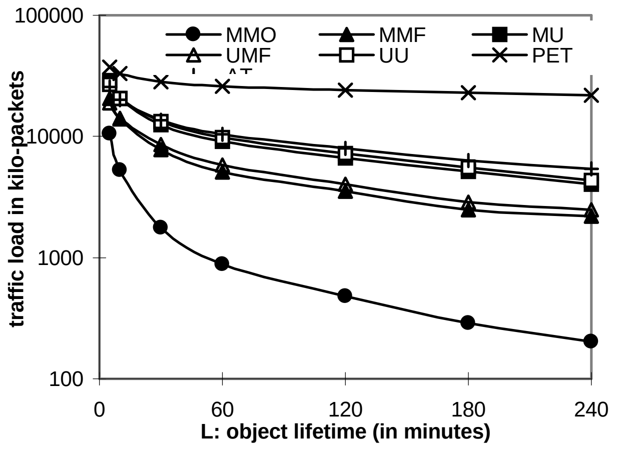

Figure 6.3 Packet Count (Surge, P=500, V=50)

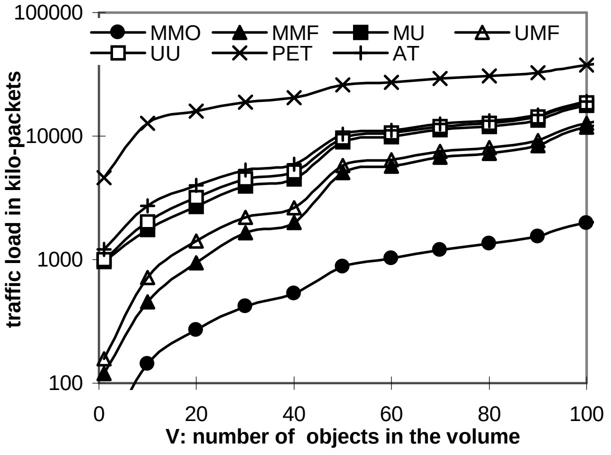

Figure 6.4 Packet Count (Surge, P=500, L=60)

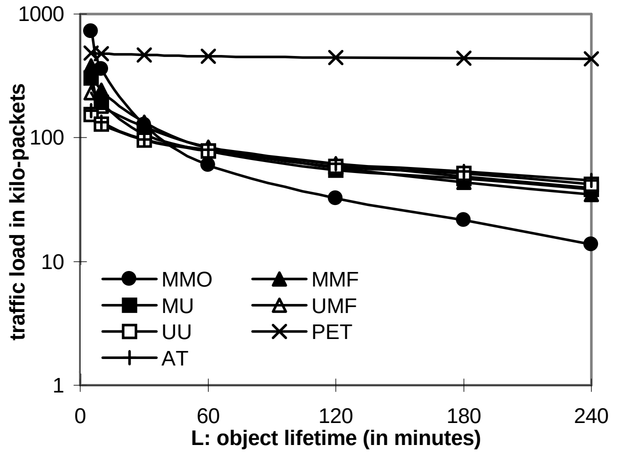

Figure 6.5 Packet Count (Stanford, P=10, V=50)

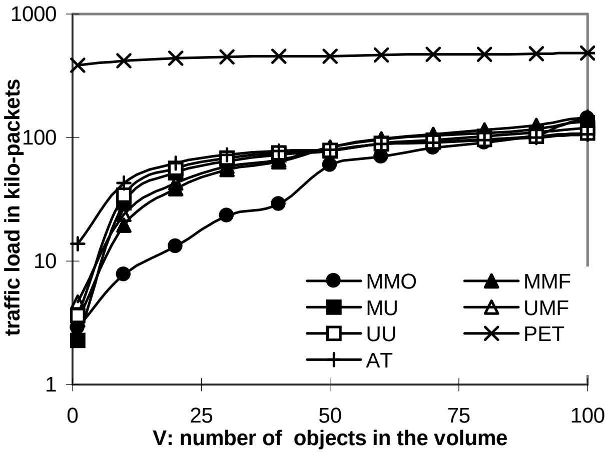

Figure 6.6 Packet Count (Stanford, P=10, L=60)

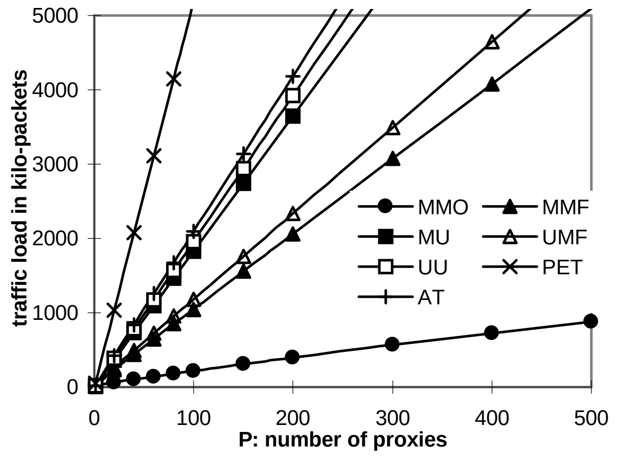

Figure 6.7 Packet Count (Surge, L=60, V=50)

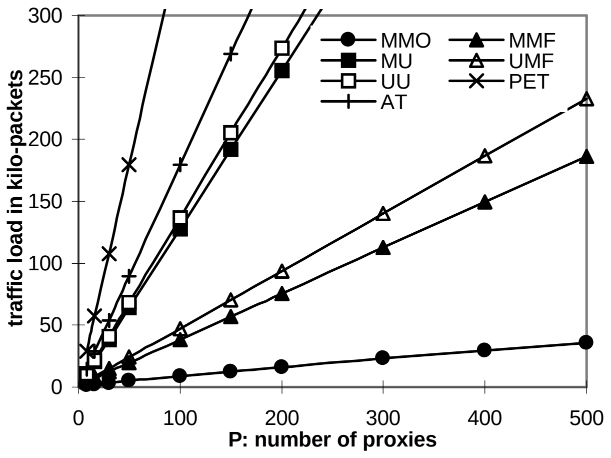

Figure 6.8 Packet Count (NLANR, L=60, V=6)

times fewer packets than TCP in order to deliver a document to 500 receivers. Therefore, MMO is less efficient than repeated AT only when over 96% of the data received is extraneous.

Figures 6.7 and 6.8 further explain the audience-size factor. Tangents of the curves follow the order:

MMO << MMF < UMF << MU < UU < AT << PET,

indicating that invalidation-based protocols are much more scalable than polling-based protocols. Moreover,

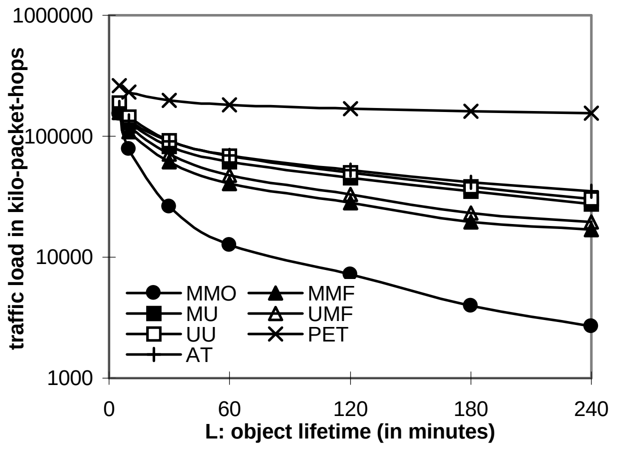

Figure 6.9 Packet-Hop Count (Surge, P=500, V=50)

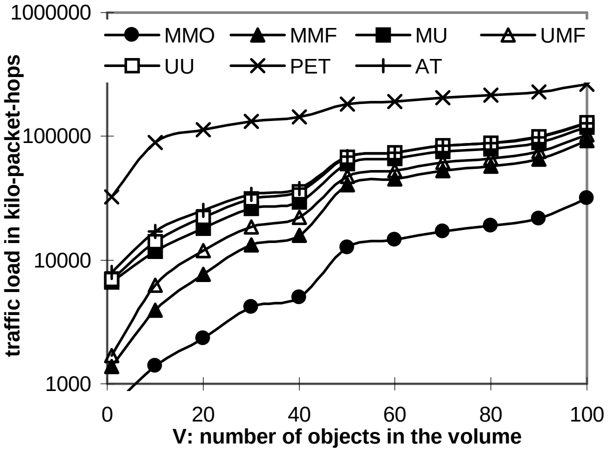

Figure 6.10 Packet-Hop Count (Surge, P=500, L=60)

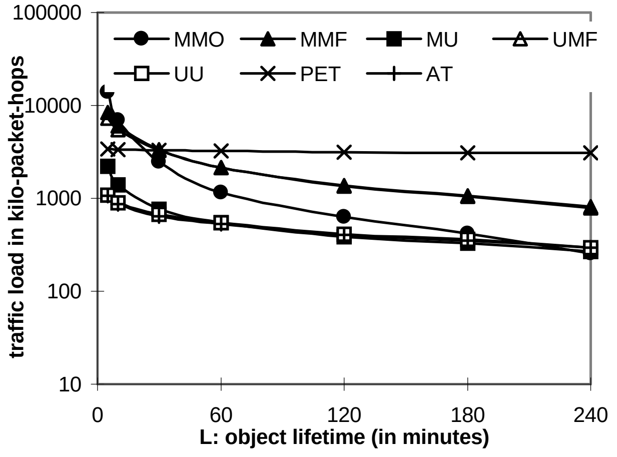

Figure 6.11 Packet-Hop Count (Stanford, P=10, V=50)

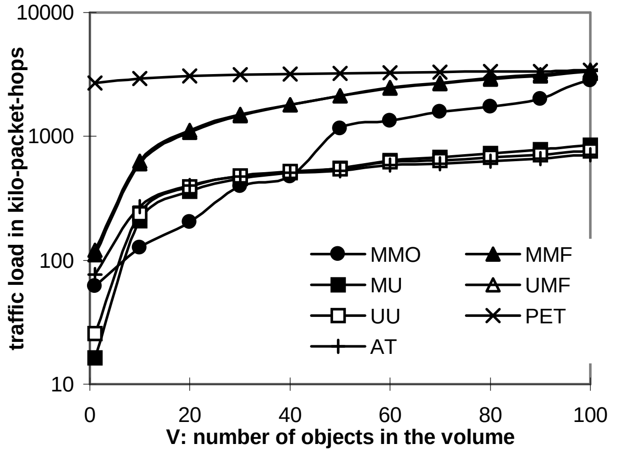

Figure 6.12 Packet-Hop Count (Stanford, P=10, L=60)

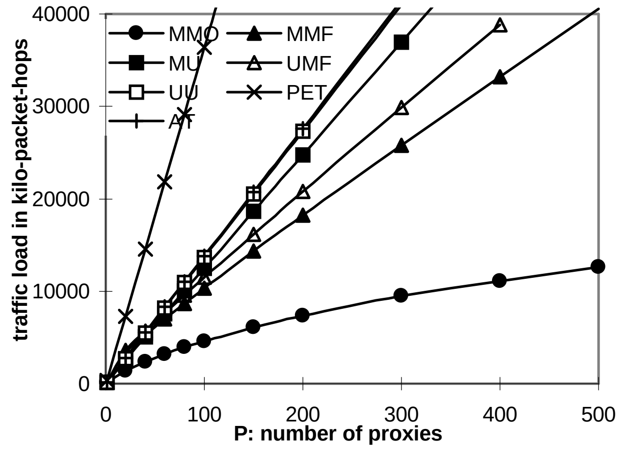

Figure 6.13 Packet-Hop Count (Surge, L=60, V=50)

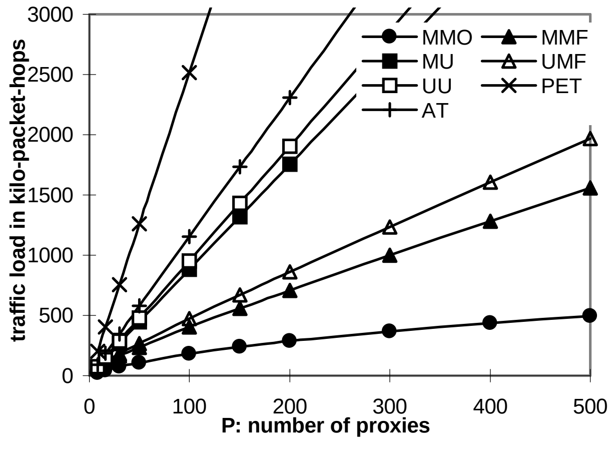

Figure 6.14 Packet-Hop Count (NLANR, L=60, V=6)

MMO is the most scalable of the invalidation-based protocols.

6.3. Network Load

Figures 6.9 through 6.14 plot the same scenarios as 6.3 to 6.8 but in packet-hop counts. Similar to the server's case, Figures 6.9 and 6.10 show that MMO is over an order of magnitude more efficient than others. With 500 proxies, multicast delivery (MMO, MMF, and UMF) always performs better than its unicast counterparts (MU and UU) and polling-based protocols (PET and

AT). Figures 6.13 and 6.14 show that MMO is far more scalable than conventional and hybrid methods from the network's perspective as well.

7. Related Work

Most related work [7, 15, 16, 29] on web caching protocols has been described in Section 3 with comparisons to MMO in that section and Section 5. Concurrent with our work, Yu et al. [37] proposed using application-level multicast for invalidations. However, their scheme presumes a pre-configured cache hierarchy in which each cache tracks web server locations and relays each HTTP miss up and down the hierarchy to the web server, and back on response. A wide flat hierarchy risks overhead from application-level routing whereas a deeper hierarchy risks latency from multiple cache hops back to the web server. This scheme, as they acknowledge, is difficult to apply with a cache mesh [3, 13, 22, 35], an emerging direction on the web. In contrast, in MMO, caches only interact as participants in a common multicast transport session; the associated subgroup hierarchy provides dynamic self-organization within a cache mesh. MMO's use of native IP multicast reduces latency and overhead on caches; its use of volumes minimizes the number of IP multicast addresses needed, addressing a key motivation of Yu [37] for going to application-level multicast. Also, the proposed EXPRESS single-source multicast [17] provides a large number of multicast addresses per server.

Another application-level technique is piggyback invalidation and validation [18,19]. However, this approach is just an optimization over unicast polling, which we have compared earlier.

Continuous multicast push (CMP) and asynchronous multicast push (AMP) [1, 26, 27] deliver popular content to end-users via native multicast. However, the server has to multicast an object continuously or many times per modification, while MMO multicasts content once per modification. Furthermore, to improve the efficiency, CMP needs to increase the amount of content carried in a multicast channel and AMP increase the wait period between two consecutive multicast deliveries, both of which prolong the end-users' web access time. Conversely, MMO reduces the web access time by always providing "fast hits" from caches.

8. Conclusion

The scalability of web caches for frequently updated objects can be significantly improved using a reliable multicast channel to proactively disseminate cache invalidations and object updates from the web server to web cache proxies. We have shown that MMO can provide fast web access, strong cache consistency, efficient bandwidth utilization and, more importantly, scalability for both the server and the network.

Considering the MMO benefits in more detail, first, the response time improves substantially for frequently updated objects (with a lifetime under 4 hours) by more than 40% over conventional caching. Second, the stale rate is reduced to zero, compared to 5% ~ 15% using a weak-consistency protocol. Even a 1% stale rate can be disastrous in applications such as medical and financial decision-making. Third, considering traffic load, MMO is over an order of magnitude more efficient than hybrid protocols, and almost two orders more than traditional ones (with 500 proxies), allowing web servers and the Internet infrastructure to meet the explosive Web growth with better service quality and lower processing and bandwidth costs.

Forming optimal volumes (so that volume objects are correlated) works better than using a separate channel for each delivery (so that proxies may choose whether or not to join the channel), in terms of reducing extraneous traffic and multicast overheads. Our experiments show that, even with random volume formation, MMO can outperform other protocols in a range of volume sizes; the range widens as the audience size grows (because of the bigger bandwidth savings over TCP). Also, the web server can form larger and better-correlated volumes based on access statistics [9]. Given a reasonably formed volume, carrying both invalidations and objects in the same channel greatly reduces the multicast session overhead as well as the address allocation and routing overhead. Conversely, our simulations find that, using a separate delivery channel, the multicast overheads can hardly be amortized over a single delivery, especially with most web objects being of small sizes.

We conclude that MMO, among the seven protocols studied, is the most efficient for disseminating popular, frequently modified and correlated objects in a volume — such as CNNfn.com or ESPN.com — to a large number of web cache proxies.

Our results to date are based on a limited set of traces. Other traces may give different quantitative results. However, we do not expect them to contradict our basic findings unless a web site hosts only highly unrelated objects.[5] The use of multicast update of cached objects

[5] The extreme is when each object is interesting to a small group of proxies and there is no overlap of interests among groups. Then no matter how the volume is constructed, either the amount of extraneous traffic is too much or the volume size and multicast group size are too small to benefit from multicast. Such objects can be disseminated via unicast.

in wide-area networks is limited in practice at present by the lack of WAN multicast support. However, as multicast is deployed in high-speed WANs to support compelling applications such as Internet TV stations, MMO is expected to become another attractive use of multicast. In fact, it completes a spectrum of delivery options for the server, from end-to-end multicast delivery for real-time video at one extreme, to multicast update of cached frequently updated objects, to unicast response to explicit requests at the other extreme. Considering this spectrum, this paper recognizes and addresses an important and growing class of objects that are less dynamic than video, yet more dynamic than can be scalably cached and kept consistent using unicast callbacks.

We hope to evaluate and refine this approach further with additional simulation and experimental deployment. One refinement is to employ delta encoding to propagate object updates [36]. In any case, our results to date indicate that this approach could play a significant role in dealing with the dramatic scaling challenges arising from the explosive growth of the Web, a growth rate that shows no sign of abating.

Acknowledgement

The authors would like to thank Paul Barford, Conrad Damon, Tim Torgenrud and the NLANR scientists for providing valuable web access traces. We also would like to thank Armando Fox, Vincent Laviano, Katia Obraczka, Craig Partridge, Shankar Ponnekanti, Chetan Rai, Jonathan Stone, and the USITS reviewers for their valuable support and comments.

References

1. Almeroth, K.C.; Ammar, M.H.; Zongming Fei; "Scalable delivery of Web pages using cyclic best-effort multicast" Proceedings IEEE INFOCOM'98 Conference on Computer Communications. April 1998. p. 1214-21 vol.3

2. Barford, P.; Crovella, M.; "Generating representative Web workloads for network and server performance evaluation" SIGMETRICS '98/PERFORMANCE'98. June 1998. Performance Evaluation Review vol.26 no.1 p. 151-60

3. Bhattacharjee, S.; Calvert, K.L.; Zegura, E.W.; "Self-organizing wide-area network caches" Proceedings IEEE INFOCOM'98 Conference on Computer Communications. April 1998. p. 600-8 vol.2

4. Bowman, C.M.; Danzig, P.B.; Hardy, D.R.; Manber, U.; Schwartz, M.F.; "The Harvest information discovery and access system" 2nd International WWW Conference. Oct. 1994. Computer Networks and ISDN Systems (Dec. 1995) vol.28, no.1-2 p.19-25

5. Byers, J. W.; Luby, M.; Mitzenmacher, M.; Rege, A.; "A digital fountain approach to reliable distribution of bulk data" ACM SIGCOMM'98 Conference. Sept. 1998. Computer Communication Review (Oct. 1998) vol.28, no.4 p. 56-67

6. Calvert, K.; Zegura, E. "GT Internetwork Topology Models (GT-ITM)" http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm

7. Pei Cao; Chengjie Liu; "Maintaining strong cache consistency in the World Wide Web" 17th International Conference on Distributed Computing Systems. IEEE Transactions on Computers (April 1998) vol.47, no.4 p. 445-57

8. Chankhunthod, A.; Danzig, P.B.; Neerdaels, C.; Schwartz, M.F.; Worrell, K.J.; "A hierarchical Internet object cache" Proc. of USENIX Annual Technical Conference. Jan. 1996. p.153-63

9. Cohen, E.; Krishnamurthy, B.; Rexford, J.; "Improving end-to-end performance of the Web using server volumes and proxy filters" ACM SIGCOMM'98, Computer Communication Review (Oct. 1998) vol.28, no.4 p.241-53

10. Fenner, W.; Casner, S. "A "traceroute" facility for IP Multicast", Internet Draft <draft-ietf-idmr-traceroute-ipm-02.txt>, November, 1997, work in progress.

11. Floyd, S.; Jacobson, V.; Liu, C.-G.; McCanne, S.; Zhang, L.; "A reliable multicast framework for light-weight sessions and application level framing" IEEE/ACM Transactions On Networking. Dec.1997. vol.5, no.6, p. 784-803

12. Gray, C.G.; Cheriton, D.R.; "Leases: an efficient fault-tolerant mechanism for distributed file cache consistency" 12th SOSP. Operating Systems Review 1989. vol.23, no.5, p. 202-210

13. Grimm, C.; Vockler, J.-S.; Pralle, H.; "Load and traffic balancing in large scale cache meshes" TERENA Networking Conference'98. Computer Networks and ISDN Systems (30 Sept. 1998) vol.30, no.16-18 p. 1687-95

14. Gunther, R.; Levitin, L.; Schapiro, B.; Wagner, P.; "Zipfs law and the effect of ranking on probability distributions" International Journal on Theoretical Physics. Feb. 1996. vol.35, no.2, p. 395-417

15. Gwertzman, J.; Seltzer, M.; "World-Wide Web cache consistency" Proceedings of USENIX Annual Technical Conference. Jan. 1996. p. 141-51

16. Gwertzman, J.S.; Seltzer, M.; "The case for geographical push-caching" Proceedings 5th Workshop on Hot Topics in Operating Systems (HotOS-V). May 1995. p. 51-5

17. Holbrook, H.; Cheriton, D. R.; "EXPRESS Multicast: an Extended Service Model for Globally Scalable IP multicast", SIGCOMM'99, August 1999, Harvard.

18. Krishnamurthy, B.; Wills, C.E.; "Piggyback server invalidation for proxy cache coherency" 7th International World Wide Web Conference. April 1998. Computer Networks and ISDN Systems (April 1998) vol.30 no.1-7 p.185-93

19. Krishnamurthy, B.; Wills, C.E.; "Study of piggyback cache validation for proxy caches in the World-Wide Web" Proceedings of the USENIX Symposium on Internet Technologies and Systems. Dec. 1997.

20. Li, D.; Cheriton, D. R.; "OTERS (On-Tree Efficient Recovery using Subcasting): a Reliable Multicast Protocol" 6th IEEE International Conference on Network Protocols

(ICNP'98). Oct. 1998. p. 237-245

21. Luby, M. et al. "Practical Loss-Resilient Codes". Proc. of the 29th ACM Symposium on Theory of Computing, 1997.

22. Melve, I.; Slettjord, L.; Bekker, H.; Verschuren, T. "Building a Web caching system-architectural considerations" Proceedings of 8th Joint European Networking Conference(JENC8). May 1997. p. 121/1-9

23. National Lab of Applied Network Research. "A Distributed Testbed for National Information Provisioning". http://ircache.nlanr.net/Cache/

24. Padmanabhan, V.N.; Mogul, J.C.; "Using predictive prefetching to improve World Wide Web latency" ACM Computer Communication Review, July 1996. vol.26, no.3, p.22-36

25. Perkins, C. "IP Encapsulation within IP", RFC 2003, October 1996.

26. Radriguez, P.; Biersack, E.W.; "Continuous multicast push of Web documents over the Internet" IEEE NETWORK. April 1998. vol.12, no.2, p. 18-31

27. P. Rodriguez, E. W Biersack, K. W. Ross "Improving the WWW: Caching or Multicast?" 1998 Web Cache Workshop. http://wwwcache.ja.net/events/workshop/papers.html

28. Rizzo, L.; Vicisano, L.; "A reliable multicast data distribution protocol based on software FEC techniques" Proceedings of Fourth Workshop on the Architecture and Implementation of High Performance Communications Subsystems - HPCC'97. June 1997. p. 115-24

29. Touch, J. "The LSAM Proxy Cache — a Multicast Distributed Virtual Cache" 1998 Web Cache Workshop. June 1998. http://wwwcache.ja.net/events/workshop/14/lsam.html

30. UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-mash.cs.berkeley.edu/ns/

31. D. Waitzman, C. Partridge and S.E. Deering, "Distance Vector Multicast Routing Protocol", RFC1075, Nov. 1988.

32. Wessels, D.; Claffy, K.; "ICP and the Squid web cache" IEEE Journal on Selected Areas in Communications. April 1998. vol.16, no.3, p. 345-57

33. Yin, J.; Alvisi, L.; Dahlin, M.; Lin, C.; "Using leases to support server-driven consistency in large-scale systems" Proceedings of 18th International Conference on Distributed Computing Systems. May 1998. p. 285-94

34. Yu, P.S.; MacNair, E.A.; "Performance study of a collaborative method for hierarchical caching in proxy servers" 7th International World Wide Web Conference. April 1998. Computer Networks and ISDN Systems (April 1998) vol.30 no.1-7 p.215-24

35. L. Zhang, S. Michel, K. Nguyen, A. Rosenstein "Adaptive Web Caching: Toward a New Global Caching Architecture" 1998 Web Cache Workshop, http://wwwcache.ja.net/events/workshop/25/3w3.html

36. Mogul, J.C.; Douglis, F.; Feldmann, A.; Krishnamurthy, B.: "Potential benefits of delta encoding and data compression for HTTP" ACM SIGCOMM 97 Conference. Computer Communication Review (Oct. 1997) vol.27, no.4 p. 181-94

37. Haobo Yu, Lee Breslau, and Scott Shenker, "A Scalable Web Cache Consistency Architecture" ACM SIGCOMM'99

Appendix A. the Web Access Traces

Surge [2] generates 500 proxy traces. Each aggregates requests from 2000 clients and lasts 15 hours. So the trace covers one million web clients. Requests are generated for 100 frequently updated objects (called *hot* objects). The number of requests for a hot object and its popularity rank follow the *zipfs law* [14]. The most popular object is accessed an average of 0.5 time per client, which is fairly conservative for web sites like CNN.com. In other words, a proxy receives 1000 requests to the most popular object and in total 6200 requests to the 100 hot objects. File sizes follow a hybrid Pareto and log-normal distribution with average 8.6 KB, standard deviation 85 KB, minimum 79 bytes and maximum 858 KB.

The Stanford trace is a 24-hour server log on December 8, 1998. After filtering out non-cacheable requests, the log contains 960,548 requests made by 42,804 clients to 97,630 files. 1% of the files are picked (uniformly across the popularity ranks) as hot objects. Popularity ranks are obtained by sorting the files based on the number of requests each file receives. Then out of every 100 files (consecutive on the sorted list), one is picked randomly as a hot object. The most popular object is accessed 51,687 times. The average file size is 24.6 KB with 1 byte minimum and 225 MB maximum. Clients are randomly partitioned into 10 groups. Requests from one group of clients form one proxy trace. The server trace is thus partitioned into 10 proxy traces.

The NLANR trace [23] consists of eight 24-hour proxy traces on December 17, 1998, the first day of the Desert Fox US military operation against Iraq. We selected CNN.com as the server site and a volume based on the prefix "*http://www.CNN.com/WORLD/meast/9812/17/iraq.strike*". There are 6 objects in the volume with average size 20.7 KB, standard deviation 8.4 KB, minimum 8.1 KB and maximum 30.7 KB. In the simulation, we scale up the number of proxies by replicating the traces.

Table A.1 shows that each NLANR proxy does not have many clients and requests to CNN.com. This is because these proxies are at the top of the *NLANR Cache Hierarchy*, each covering domains like .uk and .jp. Their clients are mostly lower-level web cache proxies. Hence the request streams are already highly filtered and reduced. Nevertheless, they represent an important part of the web caching reality.

Only the NLANR trace records the proxy response time (the time between reading the first byte of the request

Proxy name:	bo1	bo2	lj	pa	pb	sd	sv	uc
# clients:	55	57	49	39	55	36	65	48
total # requests to CNN:	2,191	2,402	3,317	4,200	6,686	7,668	5,093	2,833
# requests in the volume:	258	301	228	435	521	240	441	315

Table A.1 The number of clients and requests for NLANR proxies

Proxy name:	bo1	bo2	lj	pa	pb	sd	sv	uc
response time of a fast hit:	35	18	58	75	104	127	183	75
response time of a slow hit:	286	215	306	271	440	408	601	292
response time of a miss:	632	564	839	978	971	1,216	950	682

Table A.2 The average response time of NLANR proxies (in milliseconds).

and writing the last byte of the reply) and whether the response is a fast hit (TCP_HIT), a slow hit (REFRESH_HIT) or a miss (REFRESH_MISS). Note that TCP_MISS (a miss in the proxy's cache) does not occur in the simulation other than at the first time because any object in the volume is cached once accessed. Samples larger than 2 seconds are discarded. For each object, response times of multiple requests are averaged into one value for each response type, which the simulation then uses. Table A.2 has the response times further averaged across all objects in the volume. It shows that a fast hit offers far better response time than a slow hit or miss.

Appendix B. the Transport Protocols

TCP, OTERS and Digital Fountain are simulated on NS [30]. Ten transit-stub topologies[6] are generated by the GT-ITM internetwork topology generator [6]. Each topology has 1000 nodes, including 5 transit domains and 120 stub domains. Nodes are considered as either backbone routers or web cache proxies at the borders of their respective local area networks. Behind each node there may be hundreds or thousands of hosts that use the web caching service and are connected via ISP networks or corporate LANs. Links inside a stub domain are 100Mbps with 1ms delay. Links connecting stub and transit domains are 45Mbps with 15ms delay. Links inside a transit domain are 155Mbps with 8ms delay. Inter-transit-domain links are 155Mbps with 80ms delay. Link delays have random variations that adhere to an unbounded exponential distribution with 20% average variation. Losses are random and the link packet error rate (PER) is 1%. The multicast routing is static dense mode DVMRP [31].

For a given multicast group size, the following occurs. Each protocol's simulation is run 10 times on each of the 10 topologies with different seeds. The seed controls how receivers are randomly chosen from the 1000 nodes. Results of the 100 runs are averaged to produce the protocol's packet count and packet-hop count. Linear interpolation is used to estimate the traffic load of group sizes that are not simulated. Finally all the results are plotted in Figures 5.1 through 5.4 and fed to the web-caching simulation. The simulation is driven by a packet stream from a Constant Bit Rate (CBR) source at 100 KB/sec and 1KB packet size. In TCP, the CBR source reliably unicasts a file to each receiver. Its traffic load counts in all the data and ACK packets. Favoring TCP, we assume the connection setup and termination takes 5 packets rather than 7.

In OTERS, all the receivers join a multicast group at time 0 and start organizing the subgroup hierarchy (called the *fusion tree*). Whenever there is data in transmission, each group member sends heartbeats every 1 second (with random skews) to maintain the tree. The data transmission starts at 0.2 second, when the fusion tree is only partially formed. (This overlap causes extra reliability cost while the tree is under construction, which can happen when group members join and leave.) One notification is delivered using OTERS-NT, followed by a file transfer using OTERS-FT. The session overhead comes from the session organization packets. The per-packet cost consists of heartbeats, ACKs, NAKs, and retransmissions, along with the payload data. We also tried to extract *per-file overhead* but it turns out to be under 1% of the per-packet cost and therefore is not considered as a separate term in the traffic load model.

We simulated the Digital Fountain designed by Byers et al. [5], which uses Tornado codes [21] with $n = 2k_1$. The data transmission starts at time 0. All the receivers join the multicast group at time 0 and leave as soon as k_2 packets are received. According to [5], the average of k_2 is tuned to 5.48% over k_1. With 1% link PER, no receiver experiences over 50% losses (which is largely the case for any well connected web cache). Therefore the source always needs to send only two packets per payload packet. Packet-hop-wise, a router may continue to forward packets toward a receiver after it has left the group, until its multicast leave message reaches the router. To separate this overhead from the per-packet cost, multiple sets of results are collected for different file lengths. The session overhead and per-packet cost are then extracted from them.

[6] The picture at ftp://ftp.dsg.stanford.edu/pub/papers/ts0.gif shows an example 100-node topology. The ones used are 10 times larger.