An Integrated Architecture for The Scalable Delivery of Semi-Dynamic Web Content

Danny Dolev* Osnat Mokryn* Yuval Shavitt† Innocenty Sukhov*

* School of Engineering and Computer Science
The Hebrew University
Jerusalem 91904, Israel

†Dept. of Electrical Engineering — Systems
Tel Aviv University
Ramat Aviv 69978, Israel

Abstract

The competition on clients attention requires sites to update their content frequently. As a result, a large percentage of web pages are semi-dynamic, i.e., change quite often and stay static between changes. The cost of maintaining consistency for such pages discourages caching solutions. We suggest here an integrated architecture for the scalable delivery of frequently changing hot pages. Our scheme enables sites to dynamically select whether to cyclically multicast a hot page or to unicast it, and to switch between multicast and unicast mechanisms in a transparent way. Our scheme defines a new protocol, called httpm. *In addition, it uses currently deployed protocols, and dynamically directs browsers seeking for a URL to multicast channels, while using existing DNS mechanisms. Thus, we enable sites to deliver content to a growing number of users at less cost and during denial of service attacks, while reducing load on core links. We report simulation results that demonstrate the advantages of the integrated architecture, and its significant impact on server and network load, as well as clients delay.*

1 Introduction

The efficient and timely delivery of web content is one of the most important challenges in today's Internet industry and research community. Sites need to cut delays while delivering content to a large number of users. This need, along with the high costs of bandwidth at the Internet core, drives the large sites to use caching and content delivery networks (CDNs) [13, 4]. However, in their competition to attract users to return, sites alter content more frequently than before. Information and objects stored on web servers change quite frequently, often every few minutes (e.g., news flashes, bids, stock quotes) [14]. Recent studies suggested that caches or CDNs are responsible for not more than 50% of the content delivered to users [20, 26].

The question that arises is which type of objects change often, and what is the access pattern to highly changed objects. Breslau at al. [6] determined that page request distribution follows a Zipf-like distribution, but found a weak correlation between access frequency and the rate of change. Douglis at al. [14] found that 16.5% of resources that were accessed at least twice were modified every time they were accessed, and almost half of the text/html resources changed on each access after the first. [21] found that a large part of users requests, and over half of the repeated requests, are to modified files. Access pattern to files follows, as reported in many papers, a Zipf-like distribution [6, 14, 18, 21]. Moreover, the alpha parameter, as seen by the server, is larger than previously reported, and is in the range of 1.4 - 1.6. This implies, for instance, that just the top 2% of documents account for 90% of the accesses. They also found that the popularity of hot pages tends to be stable over time scale of days.

In this paper we target this phenomenon by presenting an architecture that enables sites to deliver frequently changing information to (an almost) unlimited number of users, in an efficient and scalable manner. Our architecture targets the top 2% hottest pages of busy sites, which account for the majority of accesses [21]. At the heart of the architecture is a dynamic distribution selection mechanism that enables the server to identify an increase in demand and to activate cyclic multicast delivery for high demand pages before the

site performance decreases. When demand abates our mechanism reverts to unicast delivery. We show how this scheme can be transparently integrated into the current web operation mode in a transparent way, requirinng a simple plug-in at the client side. We supply mechanisms based on current DNS to dynamically direct browsers seeking URLs to multicast channels. To do so, we define a new protocol specifier, called *httpm*, which replaces the http protocol specifier for the potential hot pages.

The integrated architecture model is designed in a way that requires minimal changes to current architectures, and relies mainly on existing mechanisms. For the deployment of our architecture we suggest two different schemes, which are based on existing building blocks. We use the digital fountain protocol [7] that offers an excellent tradeoff between transmission time and efficiency, for reliable data multicast. Into this protocol we integrate a feedback mechanism based on [5], which enables the site to estimate the size of a hot page multicast group. Both schemes are altered to handle the multicast of dynamically changing content. Our simulation results show that when switching from the unicast of a hot page to the multicast mechanism, the load on the server decreases significantly, while both the throughput and the goodput clients experience increase, and may arrive to 4 times the data received in the HTTP model. In the Integrated architecture clients experience a much lower delay than in the HTTP architecture. In addition, the load on core links decreases, thus enabling TCP users to receive better performance.

2 Integrated Architecture Description

The integrated WWW architecture was designed to allow a smooth transition from TCP based connections to a multicast based mechanism for uncachable hot pages. The architecture is based on a dynamic selection mechanism, which determines which of the hot pages should be multicast, based on inputs received from both the relevant TCP connection requests and UDP feedback information mechanism.

2.1 Server Side

The site is responsible for maintaining the hot page selection and multicast mechanisms. It predetermines a list of pages which are candidates for becoming hot pages. These pages are given the *httpm* protocol prefix. This list is quasi-static (e.g., Krishnan et al. [18, Fig. 12] showed that the list of popular entities is quite stable, in a medium size site they examined) and thus can be predetermined by the site's administrators offline. The list can be edited to reflect the relatively slow changes in the popularity of pages, which may demand configuring new entries or deleting existing ones at the DNS.

The dynamic selection mechanism is activated for each of the potential hot pages. It processes the number of incoming connection requests for them. Once a threshold has been crossed for one of the pages, the server begins the process of moving it to the multicast mechanism. It then notifies the DNS of the new address for the appropriate entry.

Once the page is multicast, new connection requests for it are answered with a 3xx HTTP response. This response indicates which multicast address to join, and specifies the needed plug-in. The response has to be determined as part of the HTTP protocol specification.

When the page is multicast, the server periodically estimates the size of its multicast group. When the estimation drops below a predefined threshold, the site distribution selection mechanism reverts the page from cyclic multicast back to unicast, and notifies the DNS of the new address, the server IP address. While multicast is beneficial for periods of high load, an estimation of the amount of needed resources at other times show it is less beneficial ([2]). Unnecessary edge links are loaded due to the propagation time of prune messages. This unnecessary load at edge networks impacts the performance other clients, and therefore should be avoided when possible. It is clear that at high load times, multicast is better than concurrent unicast.

Switching a page from multicast to unicast requires an overlap period, in which existing members of the group finish receiving the page while new joiners are directed to unicast. Therefore, the page is still multicast for a few more rounds, and then a diversion code, which can be recognized by the client browser, is multicast for another period. In this last step the multicast group is terminated.

2.1.1 Existing Building Blocks

For the multicast mechanism we incorporate two existing building blocks. The digital fountain [7] provides an efficient multicast mechanism. Reliability is achieved without the use of feedback messages from clients, and at minimal costs. The digital fountain requires an encoding system at the sending side, i.e., the site, and a decoding mechanism at the receiving side, i.e., the client. In the scheme, a page P_i, which consists of a set of k packets, is encoded into a set of $k+\ell$ packets, such that ℓ of which are redundant, to a total of n. All n packets are sent cyclically over time.

The second building block we use is based on the feedback control mechanism presented by [5], origi-

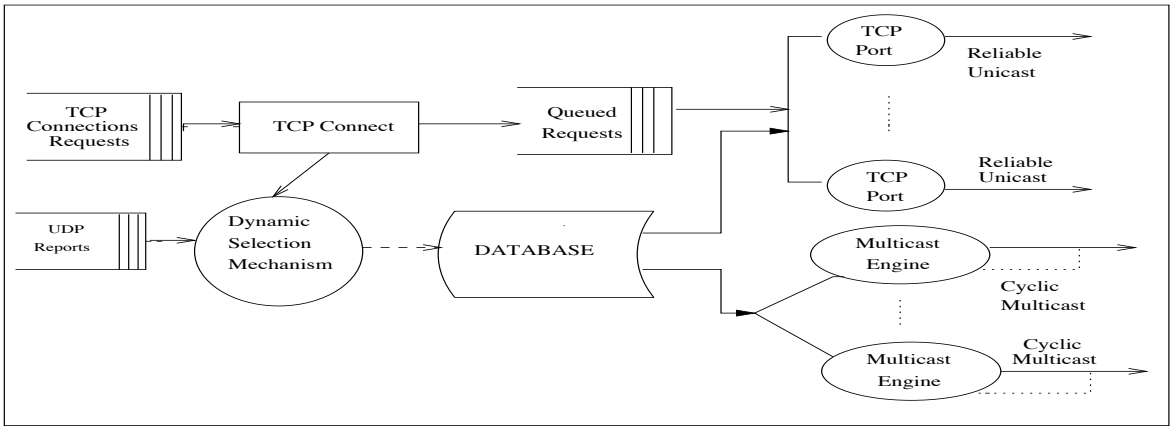


Figure 1. The Site Distribution Selection Mechanism

nally intended for multicast video distribution. It requires all participants of a multicast group to generate a random key of 16 bits. The sender sends its key, and awaits an answer from receivers with a matching key. Each period the key is sent with mask of increasing length, until an answer is obtained. Answers also contain the state of the net as perceived by receivers, for rate control mechanisms and timing.

2.1.2 Site Distribution Selection Mechanism

As can be seen in Figure 1, the dynamic selection mechanism (DSM) is part of the site's software, which determines whether hot pages are unicast or multicast. The DSM is given the following parameters: List of potential hot pages: This list is predetermined by the site's administrators. (As a result of the quasi-static nature of the list).

Unicast to multicast threshold: This threshold is actually the number of simultaneous connections that the site cannot tolerate for any one page, and would rather carry the expenses of the multicast delivery. For example, let us assume that it takes an average of 2 seconds to deliver a hot page, when the server is not overloaded and the net is not congested. (For simplicity, we assume the page requires only one connection, as is specified in HTTP1.1). If, during a minute time, the server receives 300 requests for that page, then it has to reserve 10 *simultaneous* connections throughout the minute for the delivery of this page only. We formalize this discussion in the following way: Let Δ_i be the average time estimated for any one connection to a page P_i. Let us define the number of requests to P_i in the j the period of time (typically a minute) by R_i^j. Let T_i be the connection threshold, set by the site administrators, for page P_i. Let $S_i^j = R_i^j \cdot \Delta_i$.

Then, the following conditions should be met, for the DSM to decide to multicast page P_i: *Threshold*

$$k \in [0...m] : S_i^{j-k} \geq T_i$$

Monotonically non-decreasing

$$k \in [0...m-1] : S_i^{j-k} \geq S_i^{j-k-1}$$

The range $k \in 0..m$ determines the number of periods, in which the conditions above should hold. This interval is needed to determine a pattern, rather than a momentary deviation in requests rate. For example, let $m = 3$. In this case we get that during 3 consecutive periods (minutes) there was a raise in demand for page P_i, and the demand was above the threshold, T_i;

Multicast to unicast threshold: This threshold is the number of connections that the site can tolerate for a hot page P_i. When this threshold is met during several consecutive estimations done at the DSM, and each time the size of the multicast group decreases, then the DSM switches the page from the cyclic multicast scheme to the HTTP unicast one. As discussed above, this switch is done to decrease potential unnecessary overhead encountered at the edges at low load times.

The DSM obtains the data it needs from two sources: One is the HTTP connections requests, and the other is a counting feedback mechanism. Obtaining the number of requests per hot page is rather straight forward, and involves simple lookup mechanisms. To obtain an estimation of the size of the group, a feedback mechanism based on [5] is activated. This mechanism is part of the multicast mechanism, and will be detailed there.

We assume here that as long as IPv6 is not embedded, multicast addresses constitute a precious resource, that might be rather expensive. Therefore, a site will be able to acquire, either due to the price or due to regulations, only a limited amount of such addresses, and share them between its current hot pages. Thus, even in the absence of high maintenance costs for multicast, a site will be able to use multicast only for a small subset of its pages.

2.1.3 The Multicast Mechanism

Since pages change dynamically, a page which is currently transmitted may change on disk. A major requirement in our scheme is to incorporate the change as fast as possible, as users rather have an up-to-date version, if one exists. For this purpose, the header of each multicast packet contains a *continuous* bit. When a page changes on disk, the new version of the page is multicast in the following round, and the bit is toggled. The decoder at the clients side assemblies packets with the same value in the *continuous* bit. If a change is detected, then the decoder discards all packets obtained so far, and starts assembling the packets again, until it can reconstruct the page. Since a change in content occurs only when a new round of the cyclic multicast begins, one bit is sufficient. In our suggested scheme the site uses only one channel per page. The site encodes the page for some initial loss rate (for example 10%) and sets the digital fountain's stretch factor accordingly. The encoded information is sent in a rate calculated to match the slowest client possible. Once in a few rounds, the feedback mechanism is activated. The mechanism is used to estimate both the size of the group (for the DSM) and the current loss rate in the net. Since data is transmitted at a low rate, no rate control mechanism is needed.

The packets in this round contain: The *key* field, which consists of 16 bits, and contains a randomly selected set of 16 bits; The *mask* field, which is a byte long, and specifies the number of masks bits in the key. The value of the mask field is determined by the increase in requests rate exhibited at the DSM for this page. Let R_i^M = Max R_i^j, $j \in \tau$. Then, the value of the mask field is $logR_i^M$.

If no answer was received during this round, the mask field is incremented by one, and the new mask field is sent in the next round. The process repeats until answers are obtained.

The client's decoder, upon receiving a packet containing these two fields, randomly picks a key of 16 bits. If the first *mask* digits are the same as in the site's key, it counts the exact number of packets it needs to reconstruct the page, and delivers this information back to the site, along with the read value of the *mask* field.

The site then estimates the size of the group from the number of responses obtained for a round, according to the *mask* field reported. It can also estimate the congestion along the way to the users by the number of packets received until the page could be reconstructed. If this number is bigger than the number of packets sent in a cycle, then the stretch factor is incremented, thus increment the amount of encoding information sent.

2.2 Client Side

Browsers at the client side require an additional plug-in. This plug-in identifies an *httpm* address, and acquires its IP address from the DNS. The plug-in then determines whether the address is a server's IP or a multicast address. In the case it is a multicast address, the client joins the multicast group, and activates a decoder, that decodes packets received using the digital fountain scheme. During the decoding process, control information sent along with the packets is processed, and the plug-in acts upon it. For further details, including details on *httpm* address resolution, see [12].

3 Simulation Results

In this section we show how our architecture affects performance for both clients and sites. Other works [9, 8] examined the efficiency of multicast at the network level. Our objective is to provide a first order understanding of both the throughput seen by clients and the load on the server at peak times. We use the *ns-2* [1] network simulator, which accepts Internet-like random topologies with a power law relationship of the node degree (based on [16]). We simulated a WAN network as a network of Autonomous Systems (AS's). Each AS represents either a 10 Mbps LAN or a 56Kbps dial-up line switch with active web clients. Dial-up line switches have very low connectivity. A web server is located in one of the backbone ASs, connected through a dedicated router, and supports a large number of concurrent connections. The server has seven available Web pages of different sizes and popularity. Further details on the simulation and our results that do not appear here due to lack of space can be obtained at [12].

We categorize our results by three criteria:

Goodput: The amount of data bytes sent by the server vs. the amount of data bytes received by the users. Figure 2 show the amount of data sent and received in both architectures.In order to increase the load in the Integrated model, the users request pages at 4 times the rate of users in the HTTP model. The results show, that in the HTTP model, the higher the load, relatively less data is received, up to 30% less received than sent at peak times. In the Integrated model, on the other hand, even at peak periods, the users receive up to 4 times the data sent by the server.

Server Load: The server load is obtained from sev-

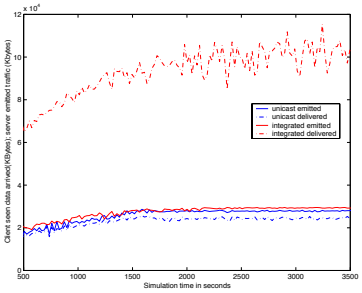


Figure 2. Increased Load Goodput: Transmitted data vs. Received data

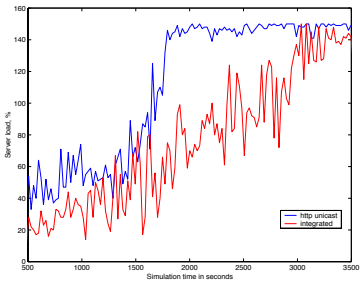


Figure 3. Server Load

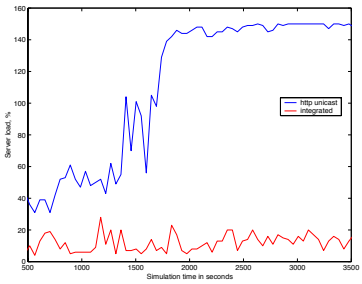
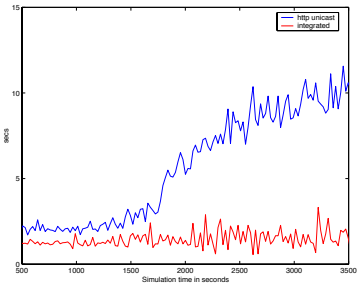


Figure 4. Server Load - Equal Delays between Clients Requests



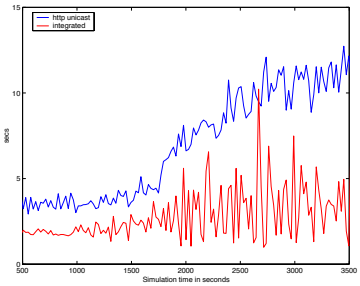


Figure 5. Client Seen Delay: pages 6 and 7

eral different statistics. First, we examine the ratio between established connections and available out channels (i.e., ports). Obviously, the server is overloaded when there are awaiting established connections in the queue, and all out channels are busy. Other criteria are the amount of dropped connections from the connection request queue, as well as the rate of establishing connections. Our results show, as can be seen in figures 3 , 4 that when the hot pages are multicast, the server load decreases dramatically.

Clients Seen Delay: We measured here the delay perceived by clients in both architectures as a function of time, while increasing the load with time until we saturate the server. The results show quite clearly that clients in the Integrated model receive these pages much faster than in the HTTP model. Figure 3 shows the amount of time it takes users in both architectures to receive pages 6 and 7, the hot pages that are multicasted in the integrated architecture. The results show quite clearly that clients in the Integrated model receive these pages much faster than in the HTTP model. Furthermore, it is beneficial to multicast these pages even if the server is not loaded. As the load increases, our results show that the delay a user suffers in the HTTP model can be 4 times the delay in the Integrated model. Additional results we obtained proved that our Integrated Architecture saves extra delay imposed by the TCP flow control mechanism, that adapts to a congested network. We also concluded, that one of the reasons for the degradation in performance seen by clients at peak times is the congestion in core links. The closer the link is to the server, the more congested it can become. Although the TCP congestion control mechanism tries to limit the effect of such peak times, by exhibiting a social behavior, the effect on both the site and the client is big. The site retransmits an extensive amount of packets until the TCP slow start mechanism takes effect, while the client experiences degradation in performance. The use of multicast mechanism for the delivery of hot pages, reduces both the traffic on core links, and the load on the server. Clients which connect by http at such times receive better service, and the amount of retransmission decreases significantly.

4 Conclusions

Frequently changing web content requires the use of new mechanisms for its scalable delivery. Our integrated architecture enables the efficient and scalable delivery of such content. Its main advantage is

its simplicity and transparency to users. Our results show that it allows sites to serve a growing amount of users at times of peak in load without experiencing performance degradation. Users delay decreases significantly, and both goodput and throughput quadruple. An important conclusion from our simulations is that, many times, performance degradation is due to increased load on core links. This increase is caused from the large amount of concurrent connections to sites, all aimed to get the exact same information. The result is congested links, which have the immediate effect of increasing the load on both the sites and the links in the short run because of retransmissions. Our scheme solves this problem by using multicast at such times, thus enabling economical use of core links.

References

[1] UCB/LBNL/VINT Network Simulator - ns (version 2), 1997. URL: http://www.isi.edu/nsnam/ns.

[2] K. Almeroth and M. Ammar. Multicast group behavior in the internet's multicast backbone (mbone). *IEEE Communications*, June 1997.

[3] K. Almeroth, M. Ammar, and Z. Fei. Scalable delivery of web pages using cyclic-best-effor (udp) multicast. *Infocom'98*, Mar. 1998.

[4] J. Angel. Caching in with content delivery. *NPN: New Public Network Magazine; http://www.networkmagazine.com*, Mar. 2000.

[5] J. Bolot, T. Turletty, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. *ACM Sigcomm'94, London, UK*, pages 58 – 67, Sept. 1994.

[6] L. Breslau, P. Cao, L. Fan, G. Philips, and S.Shenker. Web caching and zipf-like distributions: evidence and implications. *Infocom'99*, Mar. 1999.

[7] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. *Sigcomm'98, Vancouver, Canada*, Sept. 1998.

[8] R. C. Chalmers and K. C. Almeroth. Modeling the branching characteristics and efficiency gains in global multicast trees. *Infocom 2001*, April 2001.

[9] C. Chuang and M. Sirbu. Pricing multicast communication: a cost based approach. *In Proceedings of INET'98*, July 1998.

[10] R. Clark and M. Ammar. Providing scalable web services using multicast communication. *Proceedings of the IEEE Workshop on Services in Distributed and Networked Environments, Whistler, Canada*, Feb. 1995.

[11] D.Li and D.R.Cheriton. Scalable web caching of frequently updated objects using reliable multicast. *2nd USENIX Symposium on Internet Technologies and Systems (USITS)*, October 1999.

[12] D. Dolev, O. Mokryn, Y. Shavitt, and I. Sukhov. An integrated architecture for the scalable delivery of semi-dynamic web content. *School of CS and Eng., Hebrew University, Tech. report 2001-1*, Jan. 2001.

[13] A. Dornan. Farming out the web servers. *NPN: New Public Network Magazine; http://www.networkmagazine.com*, Mar. 2000.

[14] F. Douglis, A. Feldman, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: a live study of the world wide web. *In Proceedings of USENIX symp. on Internet Tech. and Systems Monterey, CA*, Dec. 1997.

[15] V. Duvvuri, P. Shenoy, and R. Tewary. Adaptive leases: A strong consistency mechanism for the word wide web. *In Proceedings of Infocom'00*, 2:834 – 843, Mar. 2000.

[16] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM*, Aug. 1999.

[17] C. Gray and D. Cheriton. Leases: An efficient fault tolerant mechanism for distributed file cache consistency. *In Proceedings of the 12th ACM Symposium on Operating Systems Principles*, pages 202–210, 1989.

[18] P. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, 8(5), Oct. 2000.

[19] C. Liu and P. Cao. Maintaining strong cache consistency in the world wide web. *In Proceedings of the ICDCS*, May 1997.

[20] J. Mogul. Squeezing more bits out of http caches. *IEEE Network magazine*, pages 6 – 14, May 2000.

[21] V. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: findings and implications. *ACM SIGCOM*, Aug 2000.

[22] G. Philips, S.Shenker, and H.Tangmunarunkit. Scaling of multicast trees: Comments on the chuang-sirbu scaling law. *In Proceedings of ACM SIGCOMM'99*, Aug 1999.

[23] P. Rodriguez and S. Sibal. Spread: Scalable platform for reliable and efficient automated distribution. *WWW9*, May 2000.

[24] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *Infocom'98*, Mar. 1998.

[25] D. Wessels. Squid internet object cache. http://www.nlanr.net/squid/. 1996.

[26] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karin, and H. Levy. On the scale and performance of cooperative web proxy caching. *In Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP99)*, pages 16 – 31, Dec. 1999.