Continuous Multicast Push of Web Documents over the Internet

Pablo Rodriguez Rodriguez, Ernst W. Biersack
Institut Eurecom
06904 Sophia-Antipolis
France
{rodrigue,erbi}@eurecom.fr

Abstract

The success of the World Wide Web has led to a steep increase in the user population and the amount of traffic on the Internet. Popular Web pages create "hot spots" of network load due to their great demand for bandwidth and increase the response time because of the overload on the Web servers. We propose the distribution of very popular and frequently changing Web documents using continuous multicast push (CMP). The benefits of CMP in the case of such documents are a very efficient use of network resources, a reduction of the load on the server, lower response times, and scalability for an increasing number of receivers. We present a quantitative evaluation of the continuous multicast push for a wide range of parameters.

1 Introduction

The exponential growth of the Internet and the World Wide Web is causing a great demand for bandwidth and server capacity. The question of how to cope with such an exponential growth does not have a simple or single answer. Scaling the Internet by simply adding more resources (bandwidth and processing power) may not suffice. Incentive schemes such as differential pricing may be part of the answer. We propose **continuous multicast push (CMP)** to use resources (server, bandwidth) more efficiently.

Popular Web pages often create "hot spots" in the network, with the same data being transmitted over the same links again and again. Currently there is a large class of document containing "short-lived" information such as news, stock market data, or auction data that, besides being of interest to a large number of receivers, change frequently. We argue that these documents can be best delivered directly to the clients using a continuous multicast distribution.

Today, we see the emergence of a new distribution paradigm on the Internet, which is called **push distribution** [1] [3] [6], where a Web server pushes the most recent version of a document to a group of subscribers. Pushing information has already been in use for many years in other areas such as television distribution, teletext, and radio distribution. However, the push mechanism implemented on the most popular Internet browsers, Netscape browser v4.02 [5] and Microsoft Internet explorer v4 [4], uses a *unicast pull-push* paradigm between the client and the source but does not exploit the benefits of a multicast distribution [30].

Multicast distribution of Web documents is an area that merits further research, given the Web's dominance of Internet traffic and the known efficiency of multicast delivery. In the case of *very popular* documents

that *change frequently*, a continuous multicast push (CMP) can offer significant benefits in terms of load reduction, bandwidth savings, reduced response times, and scalability for the growing number of receivers.

We view CMP as one of the three complementary delivery options integrated in a Web server: *Unicast, AMP, and CMP*:

- **Unicast Pull**: The document is sent as a response to a user's pull request. This delivery method is used for documents that are rarely requested.
- **Asynchronous Multicast Push (AMP)**: Requests for the same document are accumulated over a small time interval, and answered together via multicast. This delivery method is used for popular documents that have multiple requests over limited period of time.
- **Continuous Multicast Push (CMP)**: A document is continuously multicasted on the same multicast address. This delivery method is used for very popular documents that change very frequently and that are not worth caching. A Web server using CMP continuously multicasts the latest version of a popular Web document on a multicast address (every document is assigned a different multicast address). Receivers tune into the multicast group for the time required to reliably receive the document and then leave the group. In order to achieve a reliable document delivery, we propose the use of a **forward error correction code (FEC)** in combination with cyclic transmissions (see Section 3.4).

1.1 Related Work

Multicast distribution of news to news servers via the *MBONE* [20] [27] has been proposed in [28] as a way of reducing the distribution delay of articles on the *USENET* (Muse protocol). Some proposals have also been made to distribute files (operating system upgrades, images) in a local area environment via reliable multicast [18].

On the Web, one proposal to deal with popular Web documents is the use of a server-initiated caching scheme [9] [23] [12] [24]. The server sends the most popular documents towards *push-servers* near the clients. The clients get a copy of the document from the local push-server. The multicast operation could be used to periodically update the most popular documents on the push-servers. The amount of data sent can be reduced by using a delta-encoding and data-compression mechanism between the different versions of a document [34].

Clark and Ammar [17] have proposed a model where several requests for the same Web document are grouped during a certain time and answered via multicast transmission. Nonnenmacher and Biersack [36] have referred to this mode of distribution as asynchronous multicast push (AMP). Considerable bandwidth gains are achieved when the number of accumulated requests increases over a certain threshold. However, grouping requests can lead to unacceptable high response times. The AMP model is valid for the kind of information that is not delay sensitive.

Recently Almeroth, Ammar and Fei [7] have considered CMP, which they call **cyclic multicast**. They have evaluated the impact of cyclic multicast on the *service time* via a real implementation. They have also given details of how CMP could be implemented. However, their service time analysis is limited to the implementation scenario used and they have not investigated the bandwidth gain. In our proposal we have built a more general framework to analytically evaluate the gains due to CMP as a function of the document size, number of documents, request arrival rate, and network bandwidth. Therefore, we can clearly establish

the benefits of the CMP model over a wide range of parameters. Also in order to achieve reliability, we propose the use of a forward error correction code (FEC) combined with repetitive cyclic transmissions (see Section 3.4). The use of parity packets to recover from losses is more efficient and assures lower service times.

2 Caching vs Continuous Multicast Push

2.1 Caching

Caching, like CMP is another mechanism for reducing the bandwidth usage and latency to the receivers on the Internet. Caching takes place at the application layer and allows for an incremental deployment of caches. Caching is already a fact of life in much of the Internet [8]. Most **ISPs (Internet Service Providers)** and organisations connected to the Internet have been installing caches to reduce the bandwidth and decrease the latency to their users [8] [15] [10] [40] [24]. Caches can cooperate together [15] [46] [44] sharing the documents inside their caches and therefore offering higher hit rates. However, caching does not come for free and there are still open issues relating to it.

- Installing a cache requires additional resources such as computers, disks, software, etc.
- Caches need to cooperate together to increase the hit rate. This creates additional overhead.
- Caches need to maintain document consistency and provide the user with the most recent update.

In fact, the effort of installing a caching hierarchy resembles the effort that was required to put in place the first experimental multicast overlay network called **MBONE** [20] [27]. A cache hierarchy is very similar to a multicast distribution scheme but with "application hop"-by-"application hop" congestion control and reliability.

For popular Web documents that rarely change, a caching hierarchy seems the best solution. Hit rates close to 50% [8] can be achieved, and the bandwidth usage and latency to the receivers are reduced. However, there are certain Web documents that are not worth caching: news, stock market data, auction data change very fast and are of interest to many receivers. We believe that the best way to distribute these documents on the Internet is via a continuous multicast push.

The Harvest cache [15] and its public domain descendant Squid [48] are becoming the most popular caching hierarchies on the Internet [8]. In cases where the caching hierarchy stores popular and fast changing documents, each time that a request comes for an expired document, several actions take place:

1. The user request is processed at every cache level. At every cache level it is a necessary to confirm that no cache has the requested update. The request travels up the cache hierarchy until it reaches the root cache.
2. The root cache contacts the **final server** (the server with the original document) and obtains the updated document via *unicast* transmission.
3. The root cache sends the updated document via *unicast* to all the children caches that requested it. A new updated copy is placed in all the caches on the path to the receiver.

Usually the root cache of a cache hierarchy is a bottleneck [15] [40]. The root cache is heavily loaded because it deals with new document requests, updates, and missed requests that were not fulfilled by the lower level caches. The root cache in the UK cache hierarchy answers one million requests per day [46]. If the cache load is not well balanced, the latency for a receiver to obtain the updated document through the caching hierarchy may be higher than if it had contacted the final server that stores the original version of the document. A cache needs to share its load with other cooperating caches to reduce the total latency (the UK root cache consists of six machines that balance the load in a round-robin manner). Perhaps, more efficient caching schemes can be implemented but at the cost of adding complexity and more resources.

2.2 CMP

CMP takes place at the transport layer (of the OSI model) with reliability and congestion control ensured by the end systems (server and clients). In the context of the Internet, CMP requires that the network connecting a server with its clients is multicast capable: a single packet sent by a server will be forwarded along the multicast tree (see Fig 1).

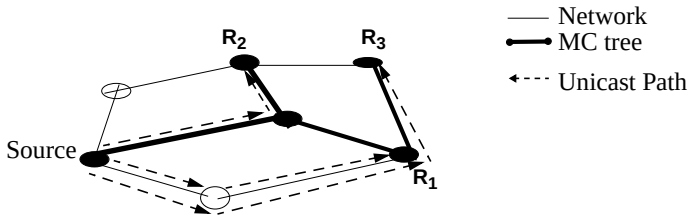


Figure 1: Network topology with multicast tree.

Where the multicast tree forks off, the multicast router will replicate the packets and send a copy on every outgoing branch of the multicast tree. Multicast routers on the Internet were first introduced via an overlay network called MBONE consisting of computers that executed the multicast routing software and that were connected via tunnels. While today the multicast routing software is part of any new router that is installed, not all the existing routers have been enabled to be multicast capable. Therefore, multicast routing on the Internet is not yet everywhere available.

A multicast distribution using CMP does not suffer problems of over-loaded servers or caches. The multicast server does not deal directly with the receivers and therefore the server complexity is much reduced (see Section 6). A multicast distribution scales very well with the number of receivers. Receivers obtain at any moment the last available update without incurring on the overhead of checking for the updated document on all the cache levels. The multicast distribution uses bandwidth efficiently by sharing *all* common paths between the source and the receivers. In the current caching hierarchy, communication is generally done via unicast between the different cache levels (there have been a number of proposals to communicate caches via multicast [49] [32]). Therefore a continuous multicast push for popular documents that change frequently seems to be the best way to deliver this kind of information on the Internet.

However, multicast distribution of Web documents on the Internet is still in its infancy as a viable service; in fact, very few network providers offer it as a service [22]. A continuous multicast distribution also requires some additional mechanisms:

- Session servers or a similar mechanism are needed to map the document's name into a multicast address.
- A Web server needs to monitor the number of document requests to decide which documents to multicast and when to stop multicasting them.
- There is an overhead in the multicast capable routers that maintain state information for each active multicast group.
- There is also an overhead due to the join and prune messages needed for the multicast tree to grow and shrink depending on the location of the receivers.
- Multicast congestion control is still an open issue.

We claim that due to the varying nature of the different Web documents, there is a room for both caching and continuous multicast distribution. It is beyond the scope of this paper to quantify when to use caching or when to use a continuous multicast push. Our paper presents and analyses a mechanism for sending frequently changing Web documents to many receivers, and establishes the benefits of CMP relative to a unicast distribution to all receivers in terms of bandwidth, latency, and load on the server.

In Section 3 we describe the CMP protocol and in Section 3.4 we analyse the reliable multicast transmission of Web documents using FEC. Section 4 evaluates the bandwidth gain from the network and Web service provider point of view. Section 5 presents the latency analysis. The load on the server is evaluated in Section 6. A summary and discussion of the results conclude the paper.

3 Continuous Multicast Push

Continuous Multicast Push of popular and frequently changing Web documents can reduce the bandwidth usage, the latency, and the load on the server: The necessary bandwidth on the Internet backbone and on the output link that connects a Web server to the Internet can be significantly reduced with CMP because common paths are shared (Section 4). The latency that the final user perceives can be reduced with CMP because the load on the server is reduced and multicast reliability can be achieved with a similar number of retransmissions as for unicast reliability (Section 5). The server's requirements and complexity can be reduced with CMP because the server does not deal directly with each receiver (Section 6).

On the other hand, a continuous multicast distribution requires some additional mechanisms (address assignment, request monitoring, congestion control); these already exist for a unicast distribution but still need further development for a multicast distribution. We propose a FEC scheme with cyclic retransmissions to achieve multicast reliability (Section 3.4). In the following subsections we present a general description of how CMP works and discuss how to implement the additional mechanisms.

3.1 General Description

Using a continuous multicast distribution to deliver popular documents that change frequently, a multicast server is able to answer many requests with the most up-to-date version of a Web document. The CMP distribution works as follows:

1. A Web server monitors the number of requests for a document to decide which documents to multicast. Only popular and frequently changing documents are sent via CMP.

2. The server takes the popular document and calculates some parity packets. Data and parities are sent cyclicly to a multicast address.

3. Receivers obtain a mapping of the document's name (URL) into a multicast address and then join the multicast group. Receivers stay in the multicast group until they have received the Web document reliably.

4. The server keeps monitoring the number of requests to stop multicasting the document if there are no more receivers.

3.2 Multicast Address Assignment

When a user requires a document, the first action is to map the document's URL to the corresponding IP address. Currently the unicast mapping is done through domain servers (DNS) that provide the unicast address of the final Web server. For a multicast case, something similar could be done: **session servers** could map the name of the requested document into the multicast address where it is being distributed.

In the current MBONE, the session directory sdr [26] [25] is frequently used to allocate addresses and ports for scheduled sessions in such a way that they do not collide. Periodically, multicast sessions are announced to a well known multicast address.

The multicast address could also be obtained from the final Web server. A client would first contact the final server via unicast. The final server itself would realize that the requested document is very popular and that it is sent through a certain multicast address, and would reply to the client with the multicast address. The client would then join that specific multicast group and retrieve the data. In this case, the join latency is increased by one RTT.

Exist solutions for implementing the URL to multicast address mapping and some concepts already developed for unicast distribution can be applied. Although it is not in the scope of this work to propose a particular solution, we believe that this first phase will not introduce significant time differences between the two models.

3.3 Monitoring Receivers' Requests

When the server starts multicasting a document, receivers simply join the multicast tree and stop dealing with the server. In this way it is impossible for the server to monitor the number of requests/sec for a document that is being multicasted, and to decide when to stop multicasting the document. In order to monitor the number of requests for a document at any moment, the server can use an estimation mechanism via feedback from receivers. Nonnenmacher and Biersack [37] have recently proposed a scalable and robust estimation scheme based on distributed timers.

Another solution is for receivers to explicitly notify the Web server after joining the multicast tree. Then, the Web server estimates how many cycles it needs to keep sending in order to satisfy all receivers given a certainty threshold [7].

3.4 Reliable CMP using FEC

IP multicast only provides multicast connectivity; it does not ensure reliable delivery. Reliable multicast over the Internet has been an area of research in recent years [29] and is currently being treated, among others, by the IRTF, the Internet Research Task Force [2]. We propose a **forward error correction code (FEC)** [35] with cyclic transmissions to achieve reliability. For a Web document fragmented into packets, the multicast sender calculates h *parity* packets for every block of k packets. For $h = 1$, the single parity can be produced via the exclusive-OR operator over the k data packets. For $h > 1$, we can use a Reed-Salomon coder [31]. The source performs cyclic transmissions of the k original packets $D_1, ..., D_k$ followed by the h parity packets $P_1, ..., P_h$ (Figure 2).

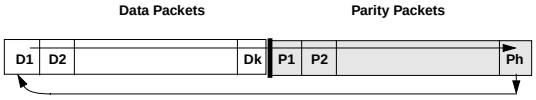


Figure 2: Arrangement of data and parities.

Receivers can join at any time. Any different k out of the $k + h$ packets sent are sufficient for decoding the original k data packets. A single parity packet can repair the loss of any data packet. The same parity packet can repair the loss of different data packets at different receivers. This means that:

- A receiver can start receiving at any time and has finished when he has received k different (data or parity) packets.
- A receiver who has successfully received $D_1, ..., D_{k-1}$ but missed D_k does not need to wait until the sender sends D_k the next time. Instead this receiver only needs one parity packet to reconstruct the missing data packet D_k. The use of parity packets will therefore reduce the time to successfully complete the reception of the document.

Compared to a scenario where only the data packets $D_1, ..., D_k$ are cyclically transmitted, sending parities reduces

- The total time until the successful reception of the whole document
- The number of data packets received unnecessarily.

If all the k data packets are received, no decoding is necessary at the receiver. The encoding/decoding costs are proportional to k and the number of parity packets produced/used [42]. Typical values of k should be such that k is between $k \in [10, 50]$ and $k + h < 255$. If the document is large, we can still keep k reasonably small by partitioning a document with $k \cdot B$ packets and organising it into B rows of k packets each (Figure 3).

On each row, for each group of k data packets, h parity packets are produced. The transmission of the rows is interleaved by sending packets in columns, one per row as shown in Figure 3. Such an arrangement imposes stricter "terminating" conditions on the receiver side. **To have finished**, a receiver not only needs at least $k \cdot B$ different packets, but the receiver must have at least k packets per row. A receiver that sees few losses can leave the multicast group much faster than a slow receiver experiencing high loss. The number of transmissions $E[M_{cmp}]$ that a single receiver needs to complete the reception of $k \cdot B$ packets is totally

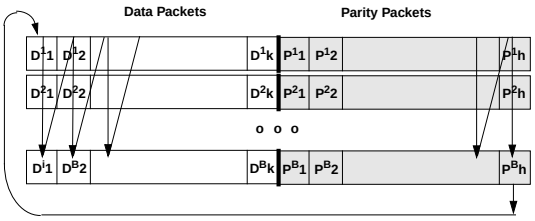


Figure 3: Partitioning of data and transmission order.

independent of the number of receivers at any moment. The continuous multicast transmission scales to many receivers, which is not the case of a unicast transmission where the Web server has to deal individually with every receiver.

We assume that packet losses occur as independent events with probability p. To simplify the analysis, we consider that all receivers are joined when the transmission of the first packet starts and that there are enough parity packets to repair all losses. If receivers join at any time and the server performs cyclic transmissions of data and parities with enough parities for a receiver to have finished before running into a second round, the analysis will be the same.

$E[M_{cmp}]$ is the average number of transmissions per packet needed to deliver $k \cdot B$ different packets to one receiver with at least k different packets per row. $E[M_{cmp}]$ is given by:

$$E[M_{cmp}] \quad = \frac{k \cdot B + E[H_{k,B}]}{k \cdot B}$$

$E[H_{k,B}]$ is the average number of parity packets transmitted before one receiver has finished. It is given by:

$$E[H_{k,B}] = \sum_{i=0}^{\infty} \sum_{j=1}^{B} H_{k,B}(i,j) \cdot P_{k,B}(i,j) \tag{1}$$

The number of parity packets $H_{k,B}$ transmitted before one receiver is able to obtain $k \cdot B$ packets with at least k different packets per row is given by:

$$H_{k,B} = \begin{cases} (i-1) \cdot B + j & i > 0 \quad 1 \leq j \leq B \\ \\ 0 & i = 0 \quad 1 \leq j \leq B \end{cases} \tag{2}$$

$P_{k,B}(i,j)$ is the probability that a receiver has finished on position (i,j) (figure 4). The fact that a receiver is able to finish at position (i,j) implies that:

- for row j, the receiver obtains k packets out of $k+i$ packets sent with packet (i,j), p_i^j
- for rows $j' < j$, the receiver obtains k packets out of $k+i$ with packet $p_i^{j'}$ or even before
- for rows $j'' > j$, the receiver obtains k packets out of $k+i$ strictly before column i.

For a single row $B = 1$, we can calculate the probability $q(i)$ that a receiver obtains k packets exactly with the reception of packet $k+i$.

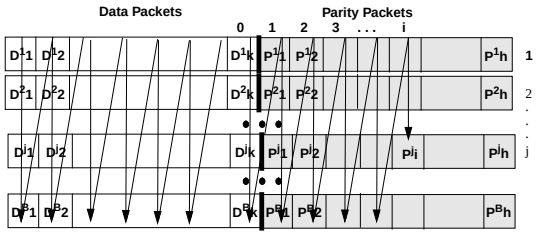


Figure 4: Receiver has finished at position (i, j).

$$q(i) = \begin{pmatrix} k+i-1 \\ k-1 \end{pmatrix} p^i \cdot (1-p)^k, \qquad i \geq 0. \tag{3}$$

The probability that a row needs $k+i$ or less packets to obtain any k packets is $Q(i)$

$$Q(i) = \sum_{t=0}^{i} q(t) \tag{4}$$

Therefore we can write the probability $P_{k,B}$ as follows:

$$P_{k,B}(i,j) = Q(i)^{(j-1)} \cdot q(i) \cdot Q(i-1)^{(B-j)} \tag{5}$$

Putting things together, we can express $E[H_{k,B}]$ in the following way

$$E[H_{k,B}] = \sum_{i=0}^{\infty} \sum_{j=1}^{B} ((i-1) \cdot B + j) \cdot Q(i)^{(j-1)} \cdot q(i) \cdot Q(i-1)^{(B-j)} \tag{6}$$

If we assume a packet size W and a document size S, the number of rows is $B = S/(k \cdot W)$. Figure 5 illustrates the behaviour of the average number of transmissions per packet $E[M_{cmp}]$ depending on the document size S for different values of k, ($k = 7, 50, 100$), a probability of loss $p = 0.01$ and a packet size $W = 1$ KB.

For a given value of k, higher document sizes S lead to higher values of $E[M_{cmp}]$. This is due to the fact that as the document size S increases, the number of rows B increases as well. When a single packet is lost on the last row $j = B$, another complete column of packets will be received before obtaining a useful parity packet that can repair the loss. The number of rows B for a given document size S decreases with increasing k. Therefore $E[M_{cmp}]$ decreases when we consider larger values of k. Even for large document sizes S and small k, the average number of transmissions per packet $E[M_{cmp}]$ is quite low.

As the block size k increases over a certain threshold, there is almost no dependence of $E[M_{cmp}]$ with the document size S. Figure 6 shows how for values of $k > 20$, $E[M_{cmp}]$ becomes very small for a wide range of document sizes S. In conclusion, we can send large documents with relatively small block sizes $k = 30$ and therefore with low coding/decoding cost, and at the same time keep the average number of transmissions low.

If the probability of loss increases, we observe from Figure 7 that for a document size $S = 1$ MB and different block sizes k, the average number of transmissions per packet $E[M_{cmp}]$ is kept low even for high probabilities of loss p.

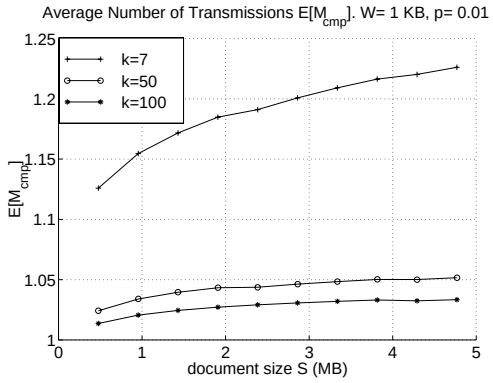


Figure 5: Average number of transmissions $E[M_{cmp}]$ versus document size S for different values of k. $W = 1$ KB, $p = 0.01$.

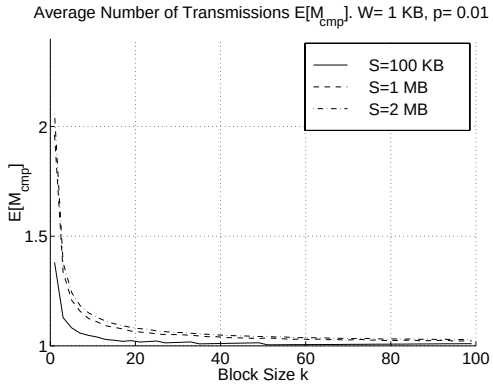


Figure 6: Average number of transmissions $E[M_{cmp}]$ versus block size k for different document sizes S. $W = 1$ KB, $p = 0.01$.

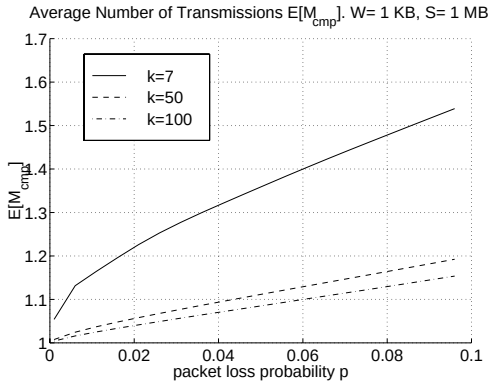


Figure 7: Average number of transmissions $E[M_{cmp}]$ versus probability of loss p for different block sizes k. $W = 1$ KB, $S = 1$ MB.

These results show that a cyclic transmission of data and parity packets can be used to send large documents with small block sizes $k = 30$ (therefore low coding/decoding costs), while keeping the average number of transmissions per packet $E[M_{cmp}]$ low over a wide range of probability of loss p.

Historically, the use of RSE codes to produce parities and to reconstruct lost data packets required special hardware to implement the RSE coder. Today this is no longer true, as has been demonstrated by Rizzo [41] with a software implementation in C of a RSE coder. Figure 8 shows the coding and decoding throughput for this coder. The measurements are made on a Pentium PC 133 running Linux with packet size $W = 1$ KB. The decoding overhead is proportional to the number of parity packets needed for reconstruction.

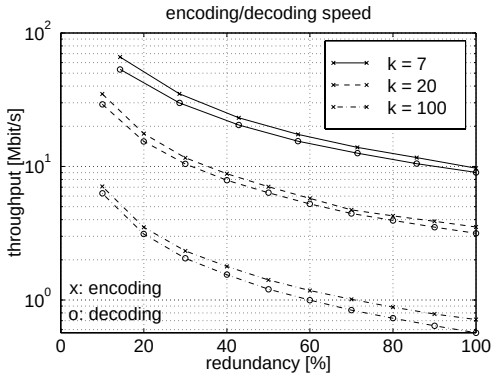


Figure 8: Coding and decoding rates in packets/sec with respect to the redundancy h/k and the transmission group size k.

Choosing $k = 30$, the time to reconstruct $k = 30$ original packets of size $W = 1$ KB from $k = 30$ parity packets is only several msec. For a typical Web document size of $S = 100$ KB, $W = 1$ KB, and $k = 30$, the number of rows B is 4. The total coding/decoding cost is the one calculated for a single row (several msec) multiplied by 4, which is still negligible compared to the global transfer time of a Web document. Therefore choosing k small, the coding/decoding times of a typical Web document are insignificant. On the other hand, for very popular documents it makes sense that parities are pre-computed.

Studies [14] [39] have shown a temporal and spatial correlation of losses in the Internet. Bolot has observed burst loss length with a mean of two [14]. In the case of burst loss, the average number of transmissions $E[M_{cmp}]$ will increase only if the burst length is higher than the number of rows B. For a unicast transmission, reliability is performed by individual retransmissions of the lost packets until they are correctly delivered to all receivers. The average number of transmissions per packet $E[M_{uc}]$ that a single unicast receiver needs to receive a packet reliably, is:

$$E[M_{uc}] \quad = \sum_{i=1}^{\infty} i \cdot p^{i-1} \cdot (1-p) = \frac{1}{1-p}$$

In Figure 9, we see the average number of transmissions for a reliable unicast and multicast distribution depending on the probability of packet loss. For the reliable multicast transmission we have taken $k = 30$ to have low coding/decoding costs while keeping $E[M_{cmp}]$ low over a wide range of probability of packet loss p.

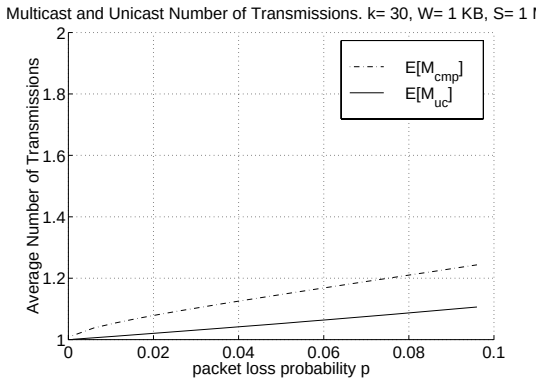


Figure 9: Average number of transmissions for reliable multicast using FEC and reliable unicast depending on the probability of loss p.

We observe that $E[M_{uc}]$ and $E[M_{cmp}]$ are very similar for a high range of probability of loss. Reliability increases the mean number of transmissions per packet for unicast and multicast transmission using FEC by a similar factor. Therefore performing reliability does not represent significant differences in terms of additional transmitted packets for a unicast transmission and a multicast transmission using FEC.

3.5 Congestion Control

Congestion control is needed to avoid a collapse in the network due to sending rates that are higher than the available network bandwidth. **Unicast congestion control** is performed by TCP via slow start and congestion avoidance. **Multicast congestion control** is more complicated because different receivers typically have different bottleneck bandwidth. It can be performed via end-to-end layered multicast [36] [47] where the source sends different parts of the data into different multicast groups (referred to as layers). Receivers adjust the delivery rate themselves by joining or dropping layers depending on the losses perceived. The source does not take part in the congestion control mechanism.

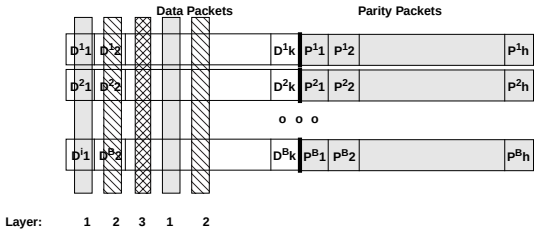


Figure 10: Layered multicast congestion control.

Figure 10 shows a multicast transmission using three layers. Column 1 of the data packets is sent on layer 1, column 2 on layer 2, column 3 on layer 3, column 4 again on layer 1 and so on. A slow receiver perceiving losses is subscribed just to layer 1. On the other hand, a fast receiver is subscribed to all the layers (1, 2, and 3). The fast receiver finishes about three times faster than the slow receiver. Whenever a receiver sees a loss, it decreases its rate by dropping layers. After a certain period without losses, the receiver increases the rate

by joining more layers. Layered multicast can deal with heterogeneous receivers and perform congestion control. However, layered multicast seems to achieve a slightly lower throughput than TCP [47] due to join and leave layer synchronisation. Layered multicast still needs some more refinements concerning the appropriate join and leave strategy for layered multicast.

3.6 General Considerations

Based on the results obtained in the previous sections, we make the following assumptions in the rest of this paper:

- The network is a lossless network. Taking into account reliability does not influence the comparison of a unicast and a multicast transmission using FEC, because reliability increases the mean number of transmissions per packet by the same factor in a unicast transmission and a multicast transmission (see Section 3.4).
- We do not explicitly model congestion control, arguing that congestion control can achieve a similar throughput in a unicast transmission via TCP and a CMP transmission via layered multicast (see Section 3.5). The transmission rates μ_{TCP} and μ_{cmp} used in the following sections for the unicast and multicast case should be considered as transmission rates achieved on the Internet, given that there are bottlenecks and that congestion control is used.

These simplifications do not have an impact on the bandwidth and latency comparison presented in the following sections.

4 Bandwidth Gain

4.1 Introduction

The fact that the available bandwidth is a scarce resource presents a challenge for its efficient use. A multicast distribution from the server to its R receivers gives great bandwidth savings compared to a unicast distribution:

- **Multicast**: Data is sent once from the server towards the R receivers following the multicast distribution tree. Data is copied inside the network whenever the multicast tree branches off.
- **Unicast**: Data is sent R times from the sender; once for every receiver.

In order to quantify the bandwidth gain obtained by a multicast distribution we have divided the **Network** into an **Internet Backbone** and several **Regional Networks** (Figure 11).

The Internet backbone is run by a **Network Provider**. The Internet backbone connects the different regional networks. Each of this regional networks is run by an Internet Service Provider (ISP). The ISP has an **output link** going into the Internet. A **Web Service Provider (WSP)** needs to rent some of the bandwidth from the ISP output link in order to run a Web server.

It is very interesting for a network provider to offer the same service using as few resources as possible from its network. A multicast distribution reduces the bandwidth costs for the network provider by sharing

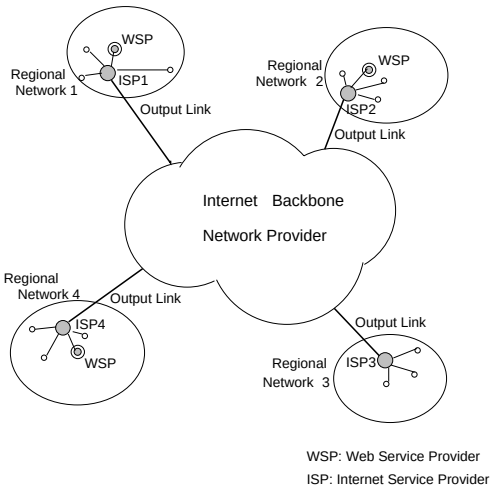


Figure 11: Network overview.

common paths. Figure 12(a) shows that in order to send a single packet to 4 receivers, a unicast distribution uses 8 links inside the Internet backbone run by the network provider. On the other hand, a multicast distribution only uses 6 links (Figure 12(b)). This gives a significant bandwidth reduction to the network provider.

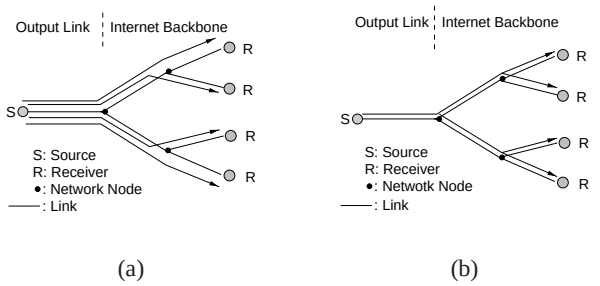


Figure 12: Unicast delivery (left) and Multicast delivery (right).

On the other hand, a Web service provider that runs a Web server needs to buy some of the bandwidth from the ISP output link into the Internet. If the Web service provider is able to answer the same number of receivers with less bandwidth, it will reduce its expenses. Figure 12 shows that the necessary output link bandwidth with a multicast distribution is 4 times less than with a unicast distribution. A multicast distribution can clearly reduce the access cost for the Web service provider.

The network provider and the Web service provider can both obtain significant cost savings due to a continuous multicast distribution of the most popular documents. Next, we analyse in detail the bandwidth gain for both points of view: Network provider and Web service provider.

4.2 Network Provider Point of View

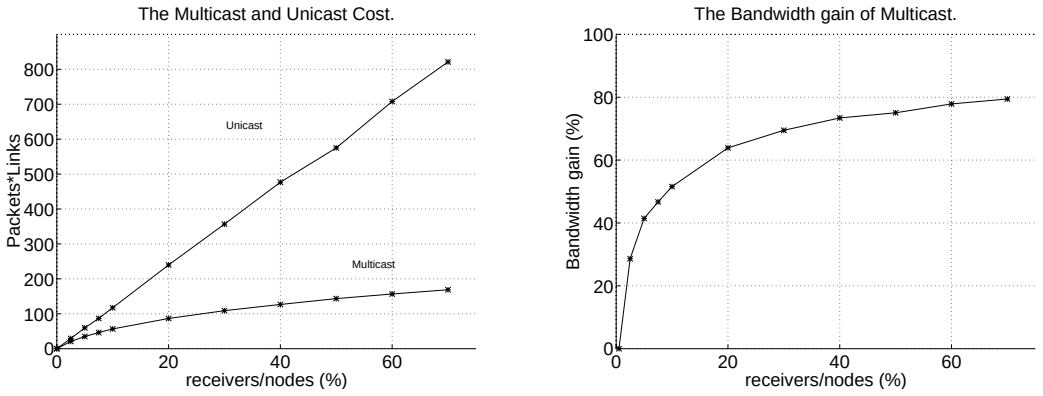
When the network provider offers a service, he is always interested in using the least possible bandwidth from his network. A multicast distribution reduces the bandwidth cost of the Internet backbone run by the network provider.

We define the **cost** as the bandwidth used on the network to deliver one packet from the source to the receivers. Every network node/link has a uniform cost per packet. The cost to deliver one packet from the source to the receivers is the product of the number of links the packet traverses and the link cost per packet. In Section 4.2.1, we deal with the case where there are no losses on the network, and in Section 4.2.2 with the case where there are losses and where therefore, reliability is needed.

4.2.1 Lossless Data Distribution

A unicast end-to-end delivery to a large number of receivers has serious limitations. For a lossless network, the results obtained by Nonnenmacher [36], show the **cost for unicast delivery** C_{uc}**, multicast delivery** C_{mc} and **lossless bandwidth gain** $G_{lossless}$ depending on the ratio of network nodes that are receivers to the network nodes (Figure 15).

$$G_{lossless} = \frac{C_{uc} - C_{mc}}{C_{uc}}$$



(a) Cost comparison between unicast and multicast.

(b) Bandwidth gain $G_{lossless}$ using multicast instead of unicast.

Figure 13: Cost comparison and Bandwidth gain.

Figure 13(a) shows that the cost for unicast delivery increases linearly with the number of receivers, while the cost for multicast delivery increases much slower with the number of receivers. When an additional receiver R' is added, the cost using a

- unicast delivery (C_{uc}) increases by the cost of the path from the source S to the receiver R'.

- multicast delivery (C_{mc}), increases by the cost of adding a few links to the multicast distribution tree to extend the multicast tree to R'.

The gain $G_{lossless}$, which expresses the cost reduction of using a multicast tree compared to multiple unicast connections, increases with the number of receivers. Figure 13(b) shows that when only 5% of the nodes are receivers, the multicast distribution reduces the bandwidth needed by 40% due to link sharing.

4.2.2 Reliable Data Distribution

The bandwidth gain $G_{lossless}$ was computed for the case of *lossless* data transmissions. If we assume that packets can be lost, we also need to include the retransmitted packets in our cost calculation. The **reliable bandwidth gain** G_{rel} is given by

$$G_{rel} = \frac{C_{uc}^{rel} - C_{mc}^{rel}}{C_{uc}^{rel}}$$

The cost $C_{uc}^{rel} = C_{uc} \cdot E[M_{uc}]$ for a *reliable unicast distribution* is the product of the number of (packets*links) C_{uc} and the average number of transmissions per packet $E[M_{uc}]$.

The cost $C_{cmp}^{rel} = C_{mc} \cdot E[M_{cmp}]$ for a *reliable multicast distribution* is the product of the number of (packets*links) C_{mc} and the average number of transmissions per packet $E[M_{cmp}]$.

As presented in Section 3.4, performing reliability adds a similar number of transmissions on a unicast and a multicast distribution. Therefore the gains G_{rel} and $G_{lossless}$(Figure 13(b)) are identical.

4.3 Web Service Provider Point of View

A Web service provider running a Web server is concerned with the access link into the Internet. A big percentage of its expenses comes from the need to rent some bandwidth from the Internet Service Provider (ISP) output link into the Internet. Therefore, it is very attractive for the Web service provider to offer the same service quality with lower bandwidth on its output access. A multicast transmission can considerably reduce the needed bandwidth on the output link. The multicast server is able to satisfy thousands of requests with a single transmission. There is no need for a high speed output link to multiplex the packets of many unicast connections.

In order to calculate the achieved bandwidth reduction, we need to characterise the distribution of the popularity between the documents inside a Web server. The high popularity of a small number of Web documents is a well-known phenomenon reported in several studies [24] [11], which have shown that *"Popular documents are very popular"* in the sense that some popular documents account for most of the requests on a Web server. Cunha, Bestavros, and Crovella [19] characterised the popularity of Web documents and confirmed the strong applicability of Zipf's law to *Web documents served by Web servers* [50]. Zipf's law states that from all documents requested from a Web server, a small number of **"hot"** documents accounts for a very high portion of all requests. The remaining **"cold"** documents only account for a few percentage of all requests. Analysing some traces from the HTTP server of the Computer Science Department at Boston University (*http://www.cs.bu.edu*) [13], Bestavros and Cunha showed that from the existing documents at this Web server, 0.5 % of all available documents accounted for about 70 % of all requests. Only 1% of all documents accounted for 90 % of all requests.

The unicast transmission of Web documents into a cache hierarchy is the most appropriate solution for "hot" documents that do not change frequently. However, for "hot" and "short-lived" documents, a multicast distribution performs better. The following analysis only considers "hot" and "short-lived" documents. When we talk of "hot" documents we always mean "hot" and "short-lived".

If "hot" and "short-lived" documents account for a high percentage of all requests, the benefits obtained by a continuous multicast push CMP are immediate. A **multicast server** would continuously multicast "hot" and "short-lived" documents and answer requests for the "cold" ones via unicast. A **unicast server** would answer all requests via unicast. To have a clear idea of the impact of using CMP instead of a unicast delivery, we consider a Web server which has a request arrival rate $\lambda = \lambda_{hot} + \lambda_{cold}$ requests/sec:

- λ_{hot} requests/sec for the **"hot group"** and
- λ_{cold} requests/sec for **"cold group"**.

The "hot group" is formed by a number N_{hot} of "hot" and "short-lived" documents that will be sent via CMP on a multicast server and via unicast on a unicast server. The "cold group" is formed by the "cold" documents that are not popular and that will be sent via unicast (on both a multicast and a unicast server) as a response to unicast pull requests from the receivers. We assume a lossless network and multicast server sending "hot" and "short-lived" documents to all receivers at a constant rate μ_{cmp} and a unicast server sending to a single receiver at the same constant rate $\mu_{TCP} = \mu_{cmp}$ (see Section 3.6).

We analyse Web servers with homogeneous and heterogeneous document sizes and different request arrival rates. We take some representative values for the document sizes to calculate the bandwidth gain from the Web service provider point of view (statistics about document sizes on the Web can be found in [16] [19]).

4.3.1 Homogeneous Document Sizes

In order to calculate the bandwidth gain offered by a CMP distribution, let us consider first the case where all documents in the "cold group" have an **homogeneous size** S_{cold} and all N_{hot} documents in the "hot group" have an **homogeneous size** S_{hot}. The output link **capacity** Cap_{uc} for a unicast server is the capacity needed to answer all requests via unicast. The output link **capacity** Cap_{cmp} for a multicast server is defined as the capacity needed to send "cold" documents via unicast and to send all N_{hot} homogeneous "hot" documents via continuous multicast push CMP (Equation 7).

$$\begin{aligned} Cap_{uc} &= \lambda_{hot} \cdot S_{hot} + \lambda_{cold} \cdot S_{cold} \\ Cap_{cmp} &= N_{hot} \cdot \mu_{cmp} + \lambda_{cold} \cdot S_{cold} \end{aligned} \tag{7}$$

The multicast server is sending all N_{hot} "hot" documents in parallel on N_{hot} different multicast addresses. The Web service provider multicasting the "hot" documents can benefit from a **bandwidth gain** G_{WSP} to reduce the necessary bandwidth on the output link given by:

$$G_{WSP} = \frac{Cap_{uc} - Cap_{cmp}}{Cap_{uc}} = \frac{\lambda_{hot} \cdot S_{hot} - N_{hot} \cdot \mu_{cmp}}{\lambda_{hot} \cdot S_{hot} + \lambda_{cold} \cdot S_{cold}}$$

Figure 14 shows the gain G_{WSP} versus the total request arrival rate λ for different λ_{hot}, with homogeneous "hot" and "cold" document sizes $S_{hot} = S_{cold} = 100$ KB.

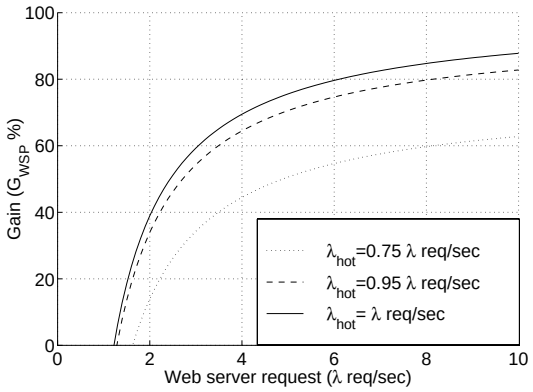


Figure 14: Web Service Provider Gain G_{WSP} versus request arrival rate λ for different λ_{hot}. $S_{cold} = S_{hot} =$ 100 KB, $N_{hot} = 10$, $\mu_{cmp} = 100$ Kbps.

For a given λ, higher values of λ_{hot} mean that the N_{hot} "hot" documents account for a higher percentage of all requests. When λ increases, the gain G_{WSP} increases as well. As λ increases, the unicast server is forced to increase its bandwidth. On the other hand, the multicast server does not need more capacity to answer more requests for popular "hot" documents, and only requires some additional bandwidth to satisfy few more requests for "cold" documents.

The comparison of a multicast server with only popular documents ($\lambda_{hot} = \lambda$) and a unicast server gives an upper bound on the gain G_{WSP} for a multicast server. If $\lambda_{hot} = \lambda$, the multicast server does not need to answer requests for "cold" documents via unicast. Therefore the gain obtained by the multicast distribution of "hot" documents is not attenuated by the unicast transmissions of "cold" documents.

If $\lambda_{hot} = 0.95\lambda$, a multicast distribution gives a reduction on the output link bandwidth up to 80 % for $\lambda = 8$ requests/sec. These results offer significant savings to the Web service provider in order to reduce its rented bandwidth.

From Figure 14, we see that for very small request arrival rates λ, the gain G_{WSP} becomes smaller than zero. This means that it is not worth multicasting the "hot" documents and it is better to deliver all documents via unicast.

It is only when the number of requests to the final server for the "hot" document is $\lambda_{hot}/N_{hot} > \lambda_{thres} = \mu_{cmp}/S_{hot}$, that a unicast distribution needs to send more packets to satisfy all receivers than the continuous multicast push. For request rates higher than **request arrival rate threshold** λ_{thres}, the multicast server obtains gains $G_{WSP} > 0$.

For a given λ_{hot} if the number of "hot" documents N_{hot} increases, the gain G_{WSP} will decrease since each "hot" document accounts for a lower number of requests λ_{hot}. Nevertheless each "hot" document is still sent at the same transmission rate μ on a multicast address. The multicast server needs more capacity to push more N_{hot} "hot" documents.

The multicast server can multicast the "hot" documents at lower transmission rates than the unicast server and the receivers will still perceive a similar latency on both the multicast and the unicast server. This is because:

- A unicast distribution greatly depends on the number of receivers. When the number of requests increases, the server becomes highly loaded, increasing the response times as well (see Section 5.3).
- On the other hand, a multicast distribution is totally independent of the number of receivers. One receiver does not experience any additional latency when new receivers join the multicast tree.

Taking the multicast transmission rate μ_{cmp} lower than the unicast transmission rate μ_{TCP}, the bandwidth gain is higher because the needed multicast capacity is smaller. Also the λ_{thres} from which G_{WSP} becomes positive, decreases. Therefore a multicast server can send at lower transmission rates than a unicast server offering a similar latency to the receivers and obtaining higher gains G_{WSP} on the output link.

"Hot" documents account for a high portion of all requests, but may account for a smaller portion of the data in bytes if the "cold" documents are very large. Therefore, it would be conceivable that the bandwidth from some busy server is mostly used to send a small number of very large documents that are "cold". As the size of the "cold" documents increases, the gain G_{WSP} decreases.

4.3.2 Heterogeneous Document Sizes

"Hot" and "short-lived" Web documents may have a very wide range of document sizes: from several hundreds of bytes such as market updates, to several MB or more. In this section, we consider the case where the "hot group" has *heterogeneous* "hot" document sizes. The "cold group" still has homogeneous "cold" document sizes S_{cold}. Let's consider first a "hot group" that has

- $N_{hot,big}$ **big "hot"** documents with a document **size** $S_{hot,big}$ accounting for $\lambda_{hot,big}$ requests/sec
- $N_{hot,typ}$ **typical "hot"** documents with a document **size** $S_{hot,typ}$ accounting for $\lambda_{hot,typ}$ requests/sec

Note that $\lambda_{hot} = \lambda_{hot,typ} + \lambda_{hot,big}$ and $N_{hot} = N_{hot,typ} + N_{hot,big}$. The multicast server sends each "hot" document again at a transmission rate μ_{cmp}. The new capacities Cap_{uc}, Cap_{cmp} for heterogeneous "hot" document sizes are given by:

$$\begin{aligned} Cap_{uc} &= \lambda_{hot,typ} \cdot S_{hot,typ} + \lambda_{hot,big} \cdot S_{hot,big} + \lambda_{cold} \cdot S_{cold} \\ Cap_{mc} &= \mu_{cmp} \cdot N_{hot} + \lambda_{cold} \cdot S_{cold} \end{aligned} \tag{8}$$

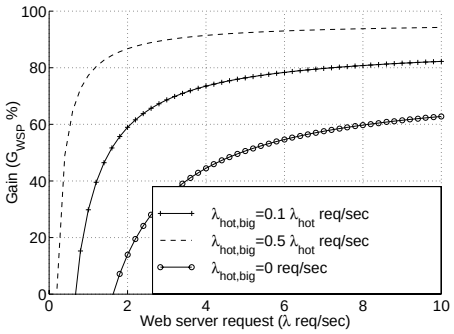
Figure 15(a) shows the gain G_{WSP} versus the total request arrival rate λ for different values $\lambda_{hot,big}$ with $S_{hot,big}$ =2 MB.

The case of $\lambda_{hot,big} = 0$ gives the gain already calculated for homogeneous "hot" document sizes. By increasing $\lambda_{hot,big}$, we are able to model more popular "hot" big documents or higher number of "hot" big documents inside the "hot" group.

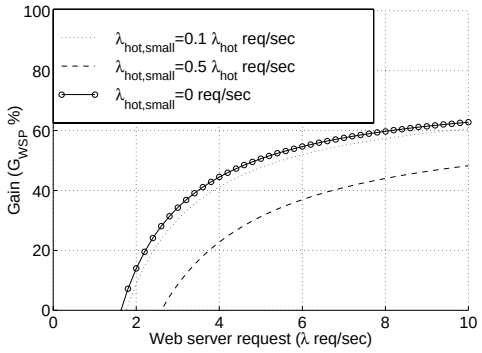
As $\lambda_{hot,big}$ accounts for a higher percentage of λ_{hot}, the gain increases significantly. If the big documents $N_{hot,big}$ account only for the 10% of λ_{hot}, the gain is increased by 20% compared to the case of homogeneous "hot" sizes for a large range of λ. We also appreciate that the request arrival rate threshold becomes smaller because the average "hot" document size being sent on the multicast server increases.

Another possible situation is a Web server where the "hot group" has:

- $N_{hot,small}$ **small "hot"** documents with a document **size** $S_{hot,small}$ accounting for $\lambda_{hot,small}$ req/sec



(a) Some big "hot" documents. $S_{hot,big}$ =2 MB.



(b) Some small "hot" documents. $S_{hot,small}$= 0.1 KB.

Figure 15: Web Service Provider Gain G_{WSP} versus request arrival rate λ for different $\lambda_{hot,small}$ and $\lambda_{hot,big}$. $S_{cold} = S_{hot,typ}$= 100 KB, N_{hot}= 10, λ_{hot}= 0.75λ, μ_{cmp} = 100 Kbps.

- $N_{hot,typ}$ *typical "hot"* documents with a document size $S_{hot,typ}$ accounting for $\lambda_{hot,typ}$ req/sec

Figure 15(b) shows the gain G_{WSP} versus the total request arrival rate λ for different $\lambda_{hot,small}$ with $S_{hot,small}$ =0.1 KB. The gain G_{WSP} is reduced compared to the case of homogeneous "hot" document sizes ($\lambda_{hot,small} = 0$). However, it is only when the $N_{hot,small}$ small "hot" documents account for 50% of all "hot" requests λ_{hot} that there is an appreciable decrease of the gain G_{WSP} by 10%. When $\lambda_{hot,small}$ increases, the request rate threshold moves slightly to higher values. The decrease on G_{WSP} having some small "hot" documents inside the "hot group" is much reduced than the increase achieved by having some big "hot" documents.

In Table 1, we have summarised the dependence of the bandwidth gain G_{WSP} and the request rate threshold λ_{thres} for the different parameters. The results obtained for some representative document sizes

	Bandwidth Gain G_{WSP}	**Arrival Rate Threshold** λ_{thres}
Homogeneous sizes (h.s.)	Higher for higher λ_{hot}	Smaller for higher λ_{hot}
Some big documents	Appreciably higher than for h.s.	Appreciably smaller than for h.s.
Some small documents	Slightly smaller than for h.s.	Slightly higher than for h.s.
Number of "hot" documents	Smaller for higher N_{hot}	Higher for higher N_{hot}
Multicast transmission rate	Higher for $\mu_{cmp} < \mu_{TCP}$	Smaller for $\mu_{cmp} < \mu_{TCP}$

Table 1: Dependence of the Web service provider gain G_{WSP} and the arrival rate threshold λ_{thres} for different parameters

show that the highest gains are achieved on a Web server that has $N_{hot,big}$ "hot" big documents that account for a high percent of all requests from the "hot" group. Also in this situation, very low request arrival rates are

already enough to obtain a gain $G_{WSP} > 0$. If the number of "hot" documents to be multicasted increases, the multicast distribution needs to continuously send to a higher number of multicast groups reducing the gain G_{WSP}. In the scenario where a Web server has some small "hot" documents, the gain and the threshold do not significantly change from the case of homogeneous document sizes, unless the small documents account for a high percentage of λ_{hot}.

5 Latency Analysis

5.1 Introduction

So far, we have seen that a significant bandwidth gain is obtained by the network provider and Web service provider using CMP. From the client point of view, the main parameter that matters is the elapsed time to obtain a document. Users do not want to wait for a long time to retrieve a document. A comparison between the latencies for our continuous multicast distribution and the actual Web pull request model is a critical measure for evaluating the feasibility and acceptability of our design.

For "hot" documents that rarely change, the best way to reduce the latency to the end user is by using a caching scheme. However, a CMP distribution is more appropiate for popular documents that frequently change.

The time to obtain a certain document can be divided into 3 different phases:

1. The time needed to map the name (URL) of the document onto a multicast group address or a unicast address (see Section 3.2).

2. The elapsed time to join the multicast tree or the equivalent time to establish a TCP connection on the unicast model.

3. The time to reliably send the required document.

5.2 Time to Join/Establish Connection

Once the client has done the mapping between the document name and the multicast address, the next step is to join the corresponding multicast group.

In a unicast distribution, a TCP connection needs to be established between the server and every receiver via a three-way hand-shake. On the other hand, in a multicast distribution, a single multicast tree is built from each server for each document distributed via CMP. The multicast tree only grows to the places where there are receivers, which are the leaves of the multicast tree.

In order to quantify the differences between the time to join the multicast tree and the time to establish a TCP connection with a unicast server, experimental data has been obtained on the MBONE. Our results [43] show that the time to establish a TCP connection is comparable to the time to join the multicast tree. If there is no receiver for a certain multicast group and a new receiver joins this group, the multicast tree has to grow from the source to the new receiver. Even in this situation, the time to join the multicast tree is similar to the time to establish a TCP connection with the unicast server. Similar results have been obtained by Almeroth, Ammar, and Fei [7]. Their results also show that the time to establish a TCP connection is comparable or even higher than the time to join the multicast tree.

5.3 Time to Transfer Data

This section analyzes the time to transfer the data reliably to the clients. We don't take into account the propagation time and only focus on the transfer time. We assume a lossless network and we do not explicitly model congestion control.

5.3.1 Unicast Transfer Time

In order to model the Web server, a simple open queue model is used. A similar approach has already been proposed by Slothouber [45]. The queueing diagram of the Web server is presented in Figure 16.

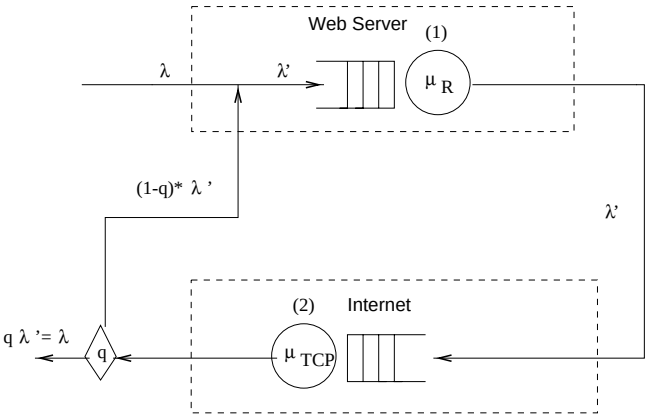


Figure 16: Queuing Network Model of a Web Server.

The model consists of two nodes; one for the Web server and one for the Internet communication. The **request arrival rate** λ is the average number of http requests received by the Web server each second. The **average document size S** is the average size of the documents served. The **server transmission rate** μ_R is the speed at which the server processes the data. The **TCP bandwidth** μ_{TCP} is an estimation of the network bandwidth, given that there are bottlenecks and there is TCP congestion control.

The time to process the requests and the initialization time is neglected because it can be simulated by a slight decrease in the server transmission rate μ_R. Initially, a job goes through the node (1) μ_R where an amount of data equal to the *MSS (maximum segment size)* is read, processed, and passed on to the network. Because modern computers can serve documents at speeds of several Mbps and the typical Internet connections (i.e. 28.8 Kbps, ISDN, T1, T2) are normally slower, the bottleneck between the server and the ouput link is usually the output link. When the bottleneck is the output link, the transfer time of the server is exclusively determined by the output link bandwidth; server transmission rate μ_R is insignificant. We will assume this situation in our approach, modeling the Web server as an output link going into the Internet (μ_R= output link bandwidth). Data travels through the Internet modeled by node (2) at transmission rate μ_{TCP}, and if the document has not been fully transmitted, the job wraps back to node (1) for further processing. This branch depends on a probability determined by the average document size S and maximun segment size MSS; the probability that the document has been fully transmitted is $q = MSS/S$. The request arrival rate λ' at node (1) is the sum of the request arrival rate λ and the request arrival rate of jobs flowing from μ_{TCP} at the exit of (2) back to (1).

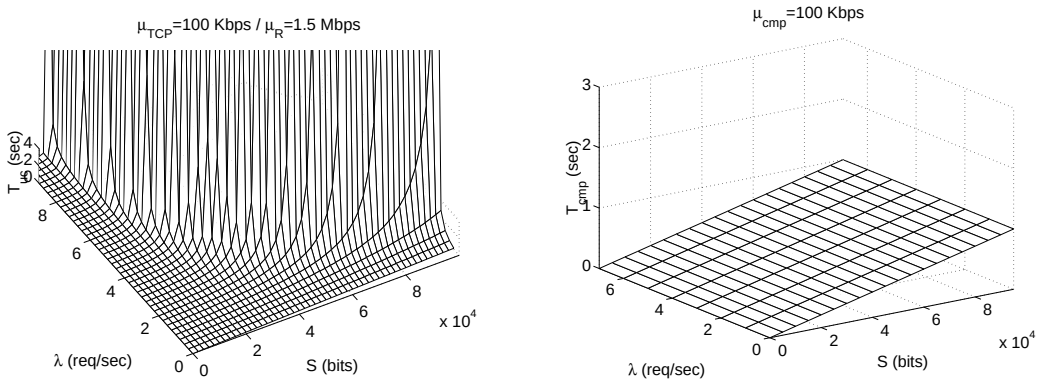
The **unicast transfer time** T_{uc} (Equation 11) of the Web model can be easily obtained from Equation 9 and Equation 10 using M/M/1 queues [33] :

$$T_{uc} = S/(\mu_{TCP} - \lambda' \cdot MSS) + S/(\mu_R - \lambda' \cdot MSS) \tag{9}$$

$$\begin{aligned} \lambda &= q \cdot \lambda' \\ q &= MSS/S \end{aligned} \tag{10}$$

$$T_{uc} = S/(\mu_{TCP} - \lambda \cdot S) + S/(\mu_R - \lambda \cdot S) \tag{11}$$

Figure 17(a) shows the transfer time T_{uc} for a Web server. We see that the time to deliver a document in a Web server increases gradually up to a point; thereafter, it increases suddenly asymptotically towards infinity defining a clear upper bound on the serving capacity. This boundary depends on the server speed μ_R, the network bandwidth μ_{TCP} and especially on the average size S and the request arrival rate λ. If the steady-state demand $\lambda \cdot S$ approaches either μ_{TCP} or μ_R, the service time T_{uc} will grow exponentially to infinity since the server is overloaded (see Figure 17(a)).



(a) Unicast Web server.

(b) Multicast Web server.

Figure 17: Unicast transfer time T_{uc} and multicast transfer time T_{cmp} versus request arrival rate λ and document size S for a unicast and a multicast Web server.

5.3.2 Multicast Transfer Time

We now turn to the case of a server continuously multicasting the requested documents. The **continuous multicast transfer time** T_{cmp} (Equation 12) depends on the multicast transmission rate μ_{cmp}, the average document size S and on the average number of transmissions per packet $E[M_{cmp}]$ (for a lossless network $E[M_{cmp}] = 1$). T_{cmp} is given by:

$$T_{cmp} = \frac{S}{\mu_{cmp}} \cdot E[M_{cmp}]$$

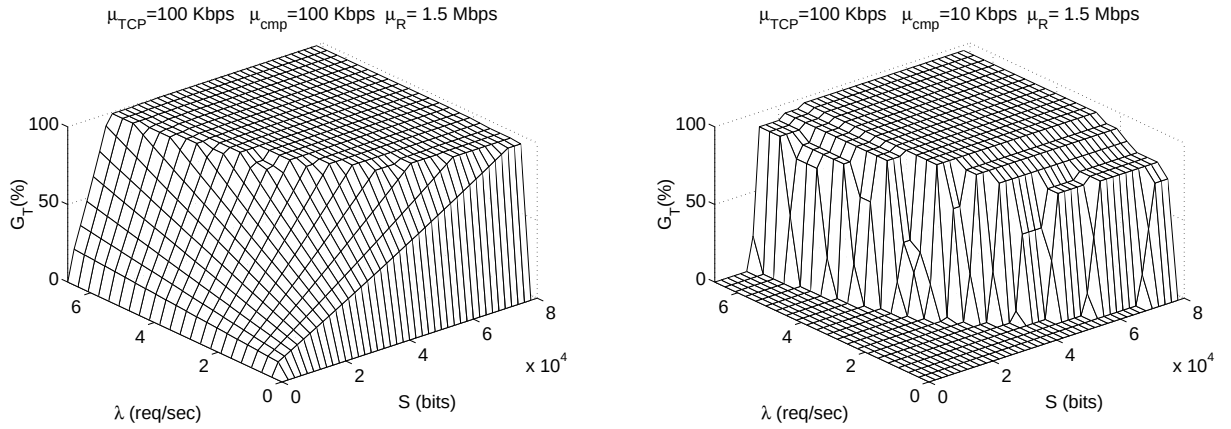
Figure 17(b) shows the transfer time for a multicast distribution T_{cmp}. The dependence with the document size S is linear and no boundary limits the capacity of the multicast server as happens on a Web unicast server. If the number of requests λ increases, T_{cmp} does not change at all because the continuously multicast transmission described in Section 3.4 is totally independent of the number of receivers R and therefore of λ. This characteristic makes the multicast distribution much more scalable than the unicast distribution. While the Web server transfer time T_{uc} goes towards infinity when λ increases, the multicast transfer time T_{cmp} remains constant for any λ.

5.3.3 Transfer Gain

In order to quantify the **transfer time gain** G_T of a multicast server over a Web server we define G_T as:

$$G_T = \frac{T_{uc} - T_{cmp}}{T_{uc}}$$

Figure 18(a) shows the transfer gain G_T for $\mu_{cmp} = \mu_{TCP}$, and Figure 18(b) for a lower multicast transmission rate $\mu_{cmp} = 0.1 \cdot \mu_{TCP}$. The Web server rate is set to output link bandwidth=μ_R= 1.5 Mbps (the output link is the bottleneck). When the unicast Web server reaches its capacity, the unicast transfer time T_{uc} goes towards infinity and therefore the transfer gain G_T goes towards 100%. We observe that a



(a) $\mu_{cmp} = \mu_{TCP}$.

(b) $\mu_{cmp} = 0.1\mu_{TCP}$.

Figure 18: Transfer gain G_T for different multicast sending rates.

$G_T = 100\%$ is obtained in both graphs for the same values of λ and S because T_{uc} is the same and only T_{cmp} changes. Even when the multicast transmission rate is ten times smaller than the unicast rate, the transfer gain is still equal to 100% for a large range of request arrival rates and document sizes. However, for very low request arrival rates and small document sizes, the transfer time for a CMP distribution becomes higher than for a unicast distribution.

In Section 4.3 we have already defined a request arrival rate threshold λ_{thres} for a given document in order to obtain positive bandwidth gains ($G_{WSP} > 0$). If the multicast transmission rate is reduced:

- λ_{thres} is smaller and the multicast server obtains higher bandwidth gains; however
- the multicast transfer time T_{cmp} becomes higher than the unicast transfer time T_{uc} for small document sizes and arrival rates.

A multicast server can save bandwidth by sending at the lower multicast transmission rates, but this will be at the expense of higher latencies to receive a document. A multicast server needs to keep the multicast transmission rate above a certain threshold $\mu_{cmp} > \mu_{thres}$ to achieve lower transfer times than the unicast distribution. Figure 19 shows the **multicast transfer rate threshold** μ_{thres} depending on the request arrival rate and the document size (note that the region where $\mu_{thres} = 0$ is not significant). The smaller the S or the

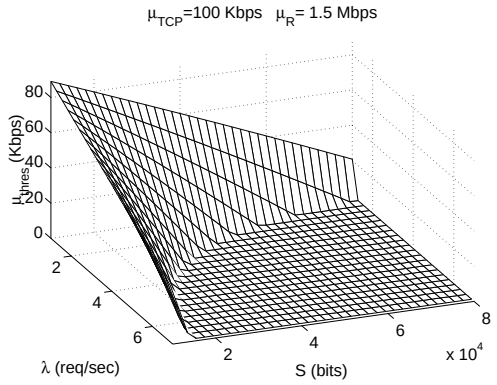


Figure 19: Multicast transfer rate threshold μ_{thres} depending on the request arrival rate λ and the document size S.

λ, the higher the threshold μ_{thres}. In all cases, we achieve a transfer gain $G_{WSP} > 0$ for a value of μ_{thres} smaller than the unicast transmission rate μ_{TCP} (which was set to $\mu_{TCP} = 100$Kbps).

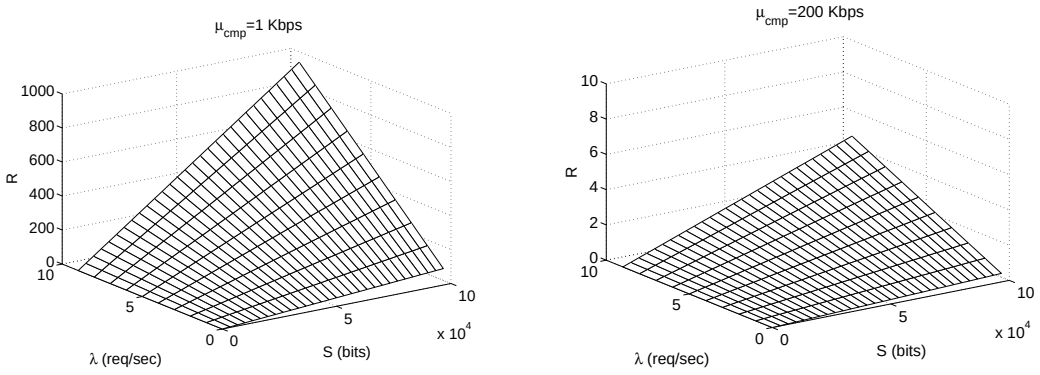
For very popular documents with high request arrival rates λ, a Web server is usually highly congested. The clients will perceive very high transfer times and some of the incoming requests will even be rejected. A multicast server on the other hand will not see its performance degrade when the request rate λ increases.

5.4 Number of Receivers

The number of receivers R in a multicast group (equation 12) depends on the document size S, the multicast transmission rate μ_{cmp}, the average number of transmissions $E[M_{cmp}]$ and the request arrival rate λ. Figures 20(a) and 20(b) give the relation between S, λ, and R for some μ_{cmp} values.

$$R = \lambda * E[M_{cmp}] * F/\mu_{cmp} \tag{12}$$

For the Internet (Figure 20(a)) with document sizes $S = 10$ Kbits and request arrival rates $\lambda = 10$ req/sec, the number of receivers $R = 1000$. For the case of an Intranet (Figure 20(b)), where the available bandwidth is higher, the same document sizes S and request arrival rates λ give a number of receivers $R = 5$. In order to get a good bandwidth gain, we know from Figure 13(b) that it is important to have a high ratio receivers/nodes(%).



(a) Number of simultaneous receivers on the Internet.

(b) Number of simultaneous receivers on an Intranet.

Figure 20: Number of receivers R versus request arrival rate λ and document size S.

6 Analysis of the Load on the Server

Another important factor in the performance of the whole system is the load on the server. In this section we try to point out the main advantages and disadvantages of both the Web server and the multicast server.

A very popular unicast server with frequently changing documents is often very loaded.

- The Web server needs to keep state information for the establishment/release of a TCP connection with every receiver. The fact that in-lined images inside a document need to establish new TCP connections places more load on the server for every Web document [38]; persistent TCP connections are allowed with the improved HTTP version: HTTP/1.1 [21].
- Due to the connection oriented nature of TCP, reliability, flow control, and congestion control are performed for every connection.
- There is also a high number of content switches to individually serve the different clients.

On the other hand, the multicast server will have lower complexity than the unicast server.

- No connection is established/released between the server and the receivers.
- The popular documents can be placed in a fast access memory, and continuously multicasted onto different multicast addresses, independently of the number and addresses of the receivers.
- Reliability is achieved using a FEC scheme [35] and cyclic transmissions. The same parity packets are able to repair different packet losses at different receivers without dealing with every receiver *individually*.
- Congestion control can be performed via layered multicast using parity packets [42] [36]. In this technique, the receivers can automatically adjust the transmission rate of delivery to themselves, reducing the sender control duties and scaling better.

In general, the requirements for a multicast server are much lower than the ones for a unicast server, especially when the number of receivers is very high. A multicast server does not need to deal with receivers individually and therefore scales very well. For the multicast distribution, reliability is performed in a more intelligent way adding parity packets to cyclic transmissions.

Nevertheless the multicast model requires some additional capability for deciding which are the popular documents to be multicasted, and for keeping track of their updates. The multicast server has also to decide when to stop multicasting a document by monitoring the number of user requests [7] [37](see Section 3.3).

7 Conclusion and Future Work

The Internet can be considered as an enormous distributed database of information. The success of the World Wide Web has contributed to a steep growth of the user population and the total traffic on the Internet. This demand requires new approaches for the distribution of "hot" documents. We have proposed a scheme that continuously delivers some popular documents that change very frequently. The key idea is to use reliable multicast transmission with a forward error correction code (FEC). We showed that delivering hot pages via continuous multicast will reduce the server load, the response times, and the network traffic.

The use of a reliable multicast transmission with FEC gives very good bandwidth gains for the network provider and Web service provider. Great latency reductions are obtained when the Web server reaches its saturation due to high request arrival rates or large document sizes. The complexity of a multicast Web server compared to a unicast Web server is clearly reduced, due to the fact that a single multicast stream can serve an arbitrary number of clients requesting the same document. These results are also valid beyond the narrow context of document delivery in the Web, and extend to any dissemination oriented application such as distribution of software, music, video or database contents.

However, CMP has also some shortcomings:

- The network needs to support multicast routing distribution.
- For each document distributed, a multicast address is used and state information is required in the routers due to the management of the multicast routing tree.

Some points still remain open: How to allocate the multicast address to which a document is being sent, the design of a complete dissemination protocol with clear rules to switch between the three different methods of distribution: *unicast, AMP, and CMP*.

In this analysis, we have limited ourselves to very popular documents that change frequently and are not worth caching. Further research is needed in order to clearly identify where to use a multicast distribution, and where to use a caching hierarchy.

8 Acknowledgements

Part of this research work was done while the first author was with EPFL (Lausanne, Switzerland). We would like to thank Jörg Nonnenmacher for his help and for the fruitful discussions that we had. Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, Hitachi, IBM France, Motorola, Swisscom, and Thomson CSF. The detailed comments of the anonymous reviewers helped considerably in improving the paper.

References

[1] "BackWeb: http://www.backweb.com/".

[2] "Internet Research Task Force (IRTF): http://www.east.isi.edu/RMRG/".

[3] "Marimba: http://www.marimba.com/".

[4] "The Microsoft White Paper about Webcasting IE 4.0: http://www.microsoft.com/ie/press/whitepaper/pushwp.htm/".

[5] "The Netscape Netcaster: http://home.netscape.com/comprod/products/communicator/netcaster.html/".

[6] "Pointcast: http://www.pointcast.com/".

[7] K. V. Almeroth, M. H. Ammar, and Z. Fei, "Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast", In *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March 1998.

[8] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "World Wide Web Caching: The Application-Level View of the Internet", *IEEE Communications Magazine*, pp. 170–178, June 1997.

[9] M. Baentsch, A. Lauer, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "Quantifying the Overall Impact of Caching and Replication in the Web", , Uni Kaiserslautern, February 1997.

[10] A. Bestavros et al., "Application-level Document Caching in the Internet", In *Proc. of SDNE'95: The second International Workshop on Services in Distributed and Network Environments*, Whistler, Canada, June 1995.

[11] A. Bestavros, "Demand-based Document Dissemination for the World-Wide Web", 95-003, Boston University, Computer Science Department, February 1995.

[12] A. Bestavros, "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems", In *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, Louisiana, USA, March 1996.

[13] A. Bestavros and C. Cunha, "Server-Initiated Document Dissemination for the WWW", *IEEE Bulletin on Data Engineering*, 1996.

[14] J. C. Bolot, "Analysis and control of audio packet loss in the Internet", In T. D. C. Little and R. Gusella, editors, *5th Workshop on Network and Operating System Support for Digital Audio and Video*, volume 1018 of *LNCS*, Springer Verlag, Heidelberg, Germany, april 1995.

[15] A. Chankhunthod et al., "A Hierarchical Internet Object Cache", In *Proc. 1996 USENIX Technical Conference*, San Diego, CA, January 1996.

[16] K. Claffy and H.-W. Braun, "Web traffic characterization: an assessment of the impact of caching documents from NCSA's web server", In *Electronic Proceedings of the Second World Wide Web Conference '94: Mosaic and the Web*, 1994.

[17] R. Clark and M. Ammar, "Providing Scalable Web Service Using Multicast Delivery", In *Proceedings of the IEEE Workshop on Services in Distributed and Networked Environments*, Whistler, Canada, June 1995.

[18] J. Cooperstock and S. Kotsopoulos, "Why Use a Fishing Line When You Have a Net? An Adaptive Multicast Data Distribution Protocol", In *USENIX 96*, 1996.

[19] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-based Traces", Technical Report 95-010, Boston University, April 1, 1995.

[20] H. Eriksson, "MBONE: The Multicast Backbone", *Communications of the ACM*, 37(8):54–60, August 1994.

[21] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, et al., "RFC 2068: Hypertext Transfer Protocol — HTTP/1.1", January 1997.

[22] A. Gove, "Stream on: RealNetwork isn't about to make multicast video a mass medium", *The Red Herring*, November 1997.

[23] J. Gwertzman, "Autonomous Replication in Wide-Area Internetworks", M.S. Thesis, Harvard, Cambridge, MA, April 1995.

[24] J. Gwertzman and M. Seltzer, "The Case for Geographical Push Caching", In *Proceedings of the Fifth Annual Workshop on Hot Operating Systems*, pp. 51–55, Orcas Island, AW, May 1995.

[25] M. Handley, "sap : Session Announcement Protocol", *Internet-Draft*, November 1996.

[26] M. Handley and V. Jacobson, "sdp : Session Description Protocol", *Internet-Draft*, November 1996.

[27] V. Kumar, "The MBONE FAQ", Collection of Information about MBONE, January 1997.

[28] J. M. Kurt Lidl, Josh Osborne, "Drinking from the Firehose: Multicast USENET News", In *proc. USENIX Winter 1994*, San Francisco, Ca, Jan 17-21 1994.

[29] B. Levine and J. J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols", *Multimedia Systems*, 1998.

[30] T. Liao, "WebCanal: a Multicast Web Application", In *Sixth International World Wide Web Conference*, april 1997.

[31] S. Lin and D. J. Costello, *Error Correcting Coding: Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[32] R. Malpani, J. Lorch, and D. Berger, "Making World Wide Web Caching Servers Cooperate", In *Fourth International WWW Conference*, Boston, Dec 1995.

[33] D. Minoli, "Broadband Network Analysis and Design", *Artech House*, pp. 116–128, 1993.

[34] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, "Potential benefits of delta-encoding and data compression for HTTP", In *Proc.SIGCOMM*, pp. 181–194, Cannes, France, September 1997.

[35] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", In *SIGCOMM '97*, pp. 289–300, Cannes, France, September 1997.

[36] J. Nonnenmacher and E. Biersack, "Asynchronous Multicast Push: AMP", In *Proceedings of ICCC'97*, pp. 419–430, Cannes, France, November 1997.

[37] J. Nonnenmacher and E. Biersack, "Optimal Multicast Feedback", In *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March 1998.

[38] V. N. Padmanabhan and J. Mogul, "Improving HTTP Latency", In *Second World Wide Web Conference '94: Mosaic and the Web*, pp. 995–1005, October 1994.

[39] V. Paxson, "End-to-End Internet Packet Dynamics", *Computer Communication Review, Proceedings of ACM SIGCOMM'97 Conference, Cannes, France, September 1997*, 27(4):139–152, October 1997.

[40] D. Povey and J. Harrison, "A Distributed Internet Cache", In *Proceedings of the 20th Australian Computer Science Conference*, Sydney, Australia, February 1997.

[41] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", *Computer Communication Review*, 27(2):24–36, April 1997.

[42] L. Rizzo and L. Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques (RMDP)", In *Proceedings of HPCS'97 Workshop*, Chalkidiki, Grece, June 1997, IEEE.

[43] P. Rodriguez, "Multicast Distribution of Web documents on the Internet", Project Report, http://www.eurecom.fr/~rodrigue/research.html, Swiss Federal Institute of Technology, Lausanne, July 1997.

[44] K. W. Ross, "Hash-Routing for Collections of Shared Web Caches", *IEEE Network Magazine*, 11, 7:37–44, Nov-Dec 1997.

[45] L. Slothouber, "A Model of Web Server Performance", In *Fifth International World Wide Web Conference*, Paris, France, May 1996.

[46] N. G. Smith, "The UK national Web cache - The state of the art", *Computer Networks and ISDN Systems*, 28:1407–1414, 1996.

[47] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer", IRTF RM Workshop September 1997, Cannes, September 1997.

[48] D. Wessels, "Squid Internet Object Cache: http://www.nlanr.net/Squid/", 1996.

[49] D. Wessels and K. Claffy, "Application of Internet Cache Protocol (ICP), version 2", Internet Draft:draft-wessels-icp-v2-appl-00. Work in Progress., Internet Engineering Task Force, May 1997.

[50] G. K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Addison-Wesley, Reading, MA, 1949.

9 Biographies

Pablo Rodriguez

Pablo Rodriguez was born in Oviedo, Spain in 1972. He obtained his M.Sc. degree as a Telecommunication Engineer from the University of Navarra UPNA, Spain in 1995. He did his Master Thesis at the department of Electrical and Electronic Engineering at King's College, London. Then, he moved to the Swiss Federal Institute of Technology EPFL, Lausanne where he did Postgraduate Studies in Communication Systems. Recently he has joined to the Institut Eurecom in Sophia Antipolis, France where he is working towards a PhD in the networking area.

His research interests are reliable multicast transmission, caching schemes for the Web and push technologies. Pablo Rodriguez is a member of the IEEE.

Ernst Biersack

Ernst Biersack received his M.S. and Ph.D. degrees in Computer Science from the Technische Universitaet Muenchen, Munich, Germany. From March 1989 to February 1992 he was a Member of Technical Staff with the Computer Communications Research District of Bell Communications Research in Morristown, USA. Since March 1992 he has been an Associate Professin Telecommunications at Institut Eurecom, in Sophia Antipolis, France. His current research is on

- Scalable Reliable Multicast Transfer for very large groups and
- Architectures for Scalable High-Performance Video Server.

In 1996, he received the outstanding paper award of the IEEE Conference on Multimedia Computing and Systems. Dr. Biersack is a member of the IEEE and ACM. He has been a program committee member for various international conferences and publicity chair for ACM SIGCOMM97.