

# Forwarding Requests among Reverse Proxies

Limin Wang<sup>†</sup>

Fred Douglass<sup>\*</sup>

Michael Rabinovich<sup>\*</sup>

<sup>†</sup>Department of Computer Science  
Princeton University  
Princeton, NJ 08544  
lmwang@cs.princeton.edu

<sup>\*</sup>AT&T Labs-Research  
180 Park Avenue  
Florham Park, NJ 07932-0971  
{douglass,misha}@research.att.com

## Abstract

Reverse proxy caching is a technology deployed by many ISPs at the border routers of their backbones to improve performance of their Web hosting services. Currently, cooperation among reverse proxies (if any) is limited to sharing each other's cache copies. In this paper, we propose to extend the cooperation by forwarding requests among cooperating reverse proxies. Instead of fetching objects from remote proxies, a proxy in this mechanism forwards requests to other proxies and tells them to send the objects directly to clients. The resulting "circular communication" (client→proxy→remote proxy→client) can be implemented in practice with a TCP hand-off among proxies.

Request forwarding has several potential benefits: first, it can get the heavy traffic off an ISP's own backbone quickly and make more backbone bandwidth available to accommodate more customers; second, it can offload busy proxies and achieve some load balancing; finally, by observing network delay of their previous interactions with clients, reverse proxies can use request forwarding to improve client-proxy network proximity and avoid congested networks. Using trace-driven simulations, we evaluated the first benefit by studying two policies. Preliminary results show a 13-35% backbone bandwidth reduction and the benefit of maintaining a dedicated output link at the content server.

## 1 Introduction

Today, 70-80% of all traffic on the Internet uses the HTTP protocol.[14] However, the tremendous growth of WWW has also resulted in a number of problems, such as overloaded servers, traffic congestion and increasing client latencies. Web caching has been recognized as an important technique to reduce Internet

bandwidth consumption [4].

By caching HTTP responses, higher throughput and lower latency can be delivered to end users. For users in Europe and Asia who have slow links to the Internet, cooperative local and regional proxy caching has already avoided many repeat requests sent to remote servers. Caching can also distribute load away from server hot-spots, and isolate end users from network failure.

Internet Service Providers (ISPs) that also provide Web hosting services often deploy *reverse proxy caching* at the border routers of their backbones. In this case, special **proxy servers** are co-located with the border routers (or *IGRs*, for Internet Gateway Routers). These proxies store HTTP responses and use them to satisfy future equivalent requests that arrive at their IGRs without going back to the end-server. By servicing content from the edge of the network, reverse proxies reduce the load on the end-servers and the backbone bandwidth consumption.

The reverse proxies can also be cooperative: on a cache miss, a proxy can obtain the requested object from another proxy that happens to have the object in its cache. While such cooperation reduces further the load on end-servers, it has limited effect on the backbone bandwidth consumption because objects sent between cooperative proxies are still sent over the ISP's own backbone. Further, current cooperative proxy caching always leaves the original proxy in charge of sending the object to the client. Thus, it does not help with any load imbalance among proxies. For the same reason, it cannot address the situation where the proxy that receives the original request is not optimal to the client in terms of network proximity or congestion.

Based on these observations, we propose that instead of fetching the missed object from a remote proxy, the original proxy send a short control message to the remote proxy and tell it to send the object *directly* to the client. By doing so, several potential

benefits can be achieved: first, cooperating reverse proxies can get the heavy traffic off their ISP’s own backbone quickly and as a result, backbone bandwidth will be saved; second, overloaded proxies can choose to forward requests to other proxies with less load to get some load balancing among all cooperative proxies; third, by observing and adapting to network traffic, forwarding requests to other proxies can be used to better utilize network proximity and avoid network congestion. In practice, this “circular” communication (client → proxy → remote proxy → client) can be achieved through a TCP hand-off among the proxies.

As the initial step in quantifying these benefits, we studied the effect of request forwarding on backbone bandwidth consumption. We performed a simulation study based on the AT&T WorldNet topology and a trace of accesses to an AT&T EasyWWW<sup>1</sup> data center and observed backbone bandwidth reduction of 13% to 35%, depending on the forwarding policy used. The other two potential benefits are still under evaluation.

The rest of the paper is organized as follows. We discuss reverse proxy caching and our motivation in more detail in the next section. We explain our approach in Section 3, and provide simulation results in Section 4. Finally, we talk about related and future work.

## 2 Motivation

### 2.1 Reverse Proxy Caching

As commercial interest in the Internet grows, more and more ISPs offer Web hosting services. They host and provide access to objects belonging to third-party information providers. Figure 1 shows a typical architecture of such service. The backbone exchanges traffic with the outside Internet only via backbone nodes that contain border routers, also known as Internet Gateway Routers (IGRs). For performance reasons, Web proxies are located at the nodes with IGRs.

When clients ask for the IP address of a web site hosted by the ISP, instead of returning the IP address of the server, the ISP’s DNS server returns the IP address of a proxy. Thus, client requests are transparently delivered to proxies, which return requested objects either from their caches or after fetching the objects from the content server(s). Since these proxies are conceptually close to and under the same ad-

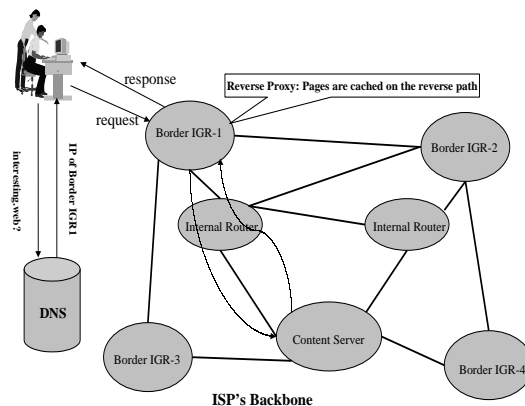


Figure 1: Reverse Proxy Caching

ministrative control as the content servers, they are often called *reverse proxies*.

For the compactness of presentation we will equate an IGR with its co-located proxy in this paper, with the understanding that in reality these are two separate devices on the same LAN.

Reverse proxies can be cooperative: when the proxy processing a request does not have the requested object but other proxies have valid copies, the first proxy can get the object from others, service the request, and cache the object. Similar to other cooperative caching, there are a number of ways for reverse proxies to track the location of cached objects:

- Probing: proxies can send Internet Cache Protocol (ICP) query messages to all fellow proxies to learn if they have a particular document [4].
- Directory: there could be a centralized directory, which holds the location information of all the proxies and all proxies may consult with it.
- Summary cache: each proxy maintains a compact summary of cache directory of every other proxy, and summaries will get updated from time to time. A query message will only be sent to those proxies whose summary suggests they could have a copy [6].

### 2.2 Problems

When a proxy obtains an object from another cooperating proxy, the object travels over the ISP’s backbone, consuming its bandwidth. Thus, proxy cooperation has only a limited effect on bandwidth savings. However, if the proxy will repeatedly serve the object

<sup>1</sup>The EasyWWW service is offered to small businesses. Typically multiple virtual web sites are supported by a single computer.

obtained from another proxy, the bandwidth overhead for obtaining the object would be amortized.

Traffic imbalance among reverse proxies can make some of them very busy, leaving others underloaded. This can lead to an inefficient resource usage within the backbone.

In addition, when a request arrives at a proxy, it is possible that another proxy is much “closer” or has a less-crowded network connection to the client. Using this “another” proxy to service the request may yield better performance to clients. Current proxy cooperation cannot utilize this advantage.

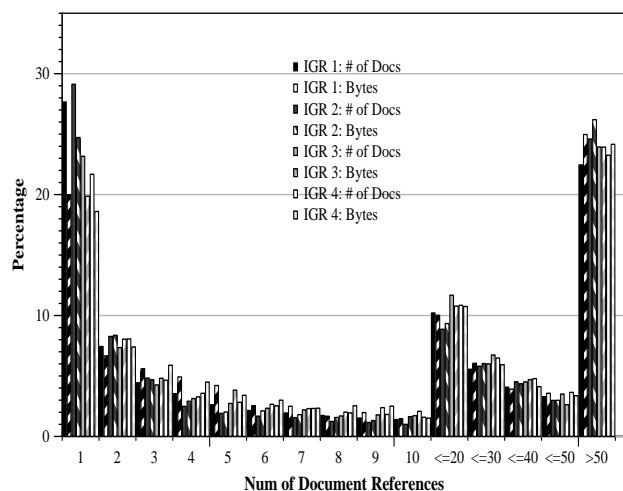


Figure 2: Histogram of IGR references to Easy WWW resources

While our proposed mechanism can potentially address all these issues, we concentrate on the issue of backbone traffic reduction in this paper. To understand how bandwidth is consumed for populating proxy caches, we analyzed the document access pattern on AT&T Worldnet IGRs using an Easy WWW trace. Figure 2 shows that the trace has two distinct peaks: more than 20% of the total number of bytes are only accessed once through any given IGR, while another 20-25% of bytes are referenced more than 50 times. All the documents accessed less than 5 times account for 40% of the documents. This interesting pattern leads to our observation that significant bandwidth savings could be obtained by reducing the bandwidth consumption for populating caches. Further, most of the documents fall into two distinct groups: one for which this bandwidth consumption is of no concern because it is amortized over many accesses, and the other which would benefit a great deal from addressing the issue.

The more bandwidth ISPs have, the more customers they can accommodate. Conversely, a

crowded backbone may result in service degradation and client complaints. Therefore our goal in this paper is to reduce the bandwidth consumption used to populate reverse proxies with cached objects. Quantifying the benefits of our approach for the other two problems mentioned above remains future work.

### 3 Our Approach

We start by describing a scheme in which proxies cooperate amongst themselves, by forwarding requests. We expand upon this scheme by permitting the requests to be forwarded directly to the content provider, providing similar benefits in cases where no other proxy has a cached copy of a page.

#### 3.1 Forwarding Requests to other Proxies

The basic forwarding approach, using cooperative reverse proxies, is depicted in Figure 3. Suppose a client sends a HTTP request to one of the border IGRs/reverse-proxies and causes a cache miss. The proxy can learn through any method we mentioned in Section 2.1 that another proxy has a fresh valid copy. Unlike the current cooperative caching where the first proxy would always get the page from the second proxy, the first proxy in our approach may decide to forward the request to the second proxy and tell it to send that object *directly* to the client. This control message can be sent through any channel, TCP or UDP etc., and will usually be much smaller compared to the response object. By forwarding requests, the heavy traffic (the actual object) leaves the backbone in “one” step, at the cost of only a small control message. If it happens that the second proxy is nearer or has less-congested connection to the client, the user may even perceive shorter latency.

Two issues must be addressed for this idea to work. First, before forwarding a request, a proxy must make sure the path from the remote proxy to the client does not wind back through the first proxy’s IGR. Otherwise, request forwarding will only add penalties without any reduction in bandwidth consumption. The proxies can use the routing information from the ISP’s own routers to avoid such a condition in the phase of selecting a remote proxy.

Second, in HTTP/1.1, persistent client connections allow multiple requests to a given server to be multiplexed or pipelined using a single TCP connection. As the objects involved could be cached on different reverse proxies, requests for these objects can be forwarded to different proxies. Still, the proper order

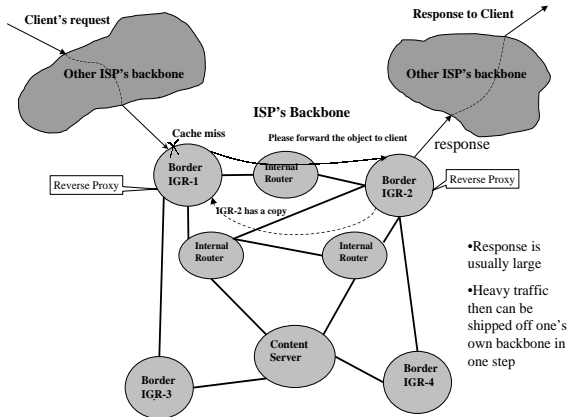


Figure 3: Forwarding Requests to Other Proxies

of the responses in the pipeline must be maintained. This suggests more synchronization among proxies. [1] addresses the similar issue in supporting P-HTTP in a cluster-based web servers. For simplicity, we did not model these issues in our simulation.

### 3.1.1 Policy Issues: Request and Proxy Selection

There are two main policy issues to consider.

- Which requests should be forwarded?
- Which remote proxy to choose for forwarding when there are multiple candidates?

#### Which Requests

Seemingly, always forwarding requests could yield the most benefit for saving backbone bandwidth. However, that is not always true. For a very hot page, forwarding control messages repeatedly may add up to higher bandwidth consumption than obtaining the page once. Also, each time a request is forwarded, the client will experience extra traversing delay from one proxy to another, if the second proxy is not much closer to the client. In contrast, after fetching the page, all future accesses would be satisfied by the receiving proxy locally. Moreover, request forwarding prevents hot pages from being replicated in multiple proxies; this may cause those proxies which keep on accepting forwarded requests for hot pages to become overloaded.

The main goal of the forwarding mechanism is therefore to select out those rarely referenced pages for bandwidth reduction. Distinguishing between “hot” pages and “cold” pages can be achieved by

maintaining a reference history of each document on every proxy. A *forwarding threshold* can be used to decide if a document is a “hot” page. Requests for seldom referenced pages are forwarded, but those for “hot” pages are filtered out for fetching.

#### Which Proxy

A simple solution to the proxy selection question is to pick a proxy that is nearest to the proxy where cache miss happens. Since this is a “local” decision within the ISP’s backbone, it is easy to implement in practice. An alternative is to choose those proxies that are nearer to the client such that user perceived latency could also be reduced. However, depending on the environment, it is likely difficult to obtain clients’ geographical location information.

In summary, forwarding requests can reduce backbone bandwidth consumption, but it may possibly increase client latency as well. The tradeoff between bandwidth and latency is the major factor affecting the policies.

Although our mechanism is targeted at reducing backbone bandwidth for populating the caches, it can also be applied in other situations, such as proxy overload or network congestion. However, how to coordinate proxies to obtain other benefits is not covered in this paper.

### 3.1.2 Implementation Issue: TCP Handoff

Our forwarding mechanism actually involves a triangular communication. Figure 4 depicts the situation: the client, the first proxy it contacts and the second proxy which sends the page directly to it form a triangle. Implementing this idea needs certain changes to the underlying software. We propose to achieve this by using a TCP handoff from the first proxy to the second [7]. First, the client sends a HTTP request to the first proxy over a normal TCP connection. Once the first proxy finds out it is a cache miss and the second proxy has a valid copy, it forwards this request to the second proxy and lets it send the response directly. At this time, the first proxy sends the second proxy a control message, which includes the state of this TCP connection, such as IP address and port number of both parties, sequence number, window size etc..<sup>2</sup> When the second proxy gets this control message, it sends the page directly to the client by injecting TCP packets addressed to the client into the network. To ensure the client will accept these

<sup>2</sup>Actually the control information needs not to be a separate message; it can be wrapped as TCP option fields, which will further reduce bandwidth consumption.

packets, the second proxy labels them as from the first proxy, in effect impersonating the first proxy to the client. The client will actually have only a half-connection with the first proxy because the response objects are from the second proxy. However, this half-connection is transparent to the client, so the TCP ACK packets from the client (which are only meaningful to the second proxy) will continue flowing to the first proxy. After the TCP handoff, the first proxy still needs to forward the TCP ACK packets from the client to the second proxy.

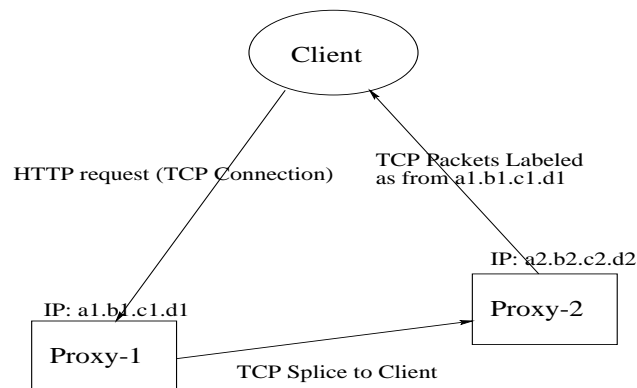


Figure 4: Implementing Triangular Communication

Obviously, this TCP handoff requires changes to proxy server software and to the TCP/IP stacks on both proxies. While we have yet to implement our approach, there are existing technologies that in different contexts have successfully implemented all the essential elements involved. IBM Network Dispatcher [7] implements handing off TCP connection from a switch to a Web server and TCP ACK forwarding. Transparent Web proxies implement the dynamic impersonation of IP addresses of Web servers to clients. LARD [12] uses a similar TCP handoff mechanism to implement request distribution for load balancing among clustered web servers. TCP splicing [9] also shares some similarity. The work by Bestavros, et al., on distributed packet rewriting [3] appears to be closest to our needs.

### 3.2 Forwarding Requests to Server

Initial simulations of the scheme in which reverse proxies forward among themselves showed only moderate benefits, as the next section will explain. Therefore, we expand on this approach with an extension that makes the content server or hosting center capable of accepting forwarded requests and sending objects to clients directly. Although clients are not allowed to contact the content server directly, at least

for the direction from client to content server, we can still set up a **dedicated output-only link** at the content server. If none of the fellow proxies has a copy, a proxy will forward the requests to the content server, which sends the response directly. The dedicated link to the rest of the Internet alleviates traffic on the backbone. This approach is depicted in Figure 5.

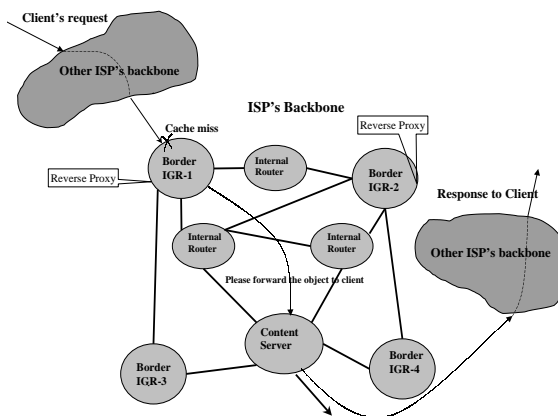


Figure 5: Let Server Send Directly

## 4 Simulation and Performance

We have simulated the request forwarding mechanism and evaluated the two schemes for backbone bandwidth reduction, the basic scheme that forwards requests among reverse proxies and the expanded scheme that includes the content server into the candidates for request forwarding. Our simulation uses a trace of accesses to servers hosted by EasyWWW, AT&T's hosting service for small businesses.

### 4.1 Simulation Model

The EasyWWW trace covers a three month period, containing about 110 million requests to more than 100 sites and 228 thousand documents. Among all the requests, 91% are from outside clients; the rest are from clients who obtain their connectivity from AT&T. The documents involved have an average size of 13KB, and 81% of them are cacheable.<sup>3</sup>

For simulation, we used the 14-node Worldnet backbone topology. Among these nodes, one is the centralized hosting center (content server); and four

<sup>3</sup>We consider URLs containing a substring of "cgi" or "?" as non-cacheable.

are border routers (or *IGRs*) through which external traffic enters the Worldnet backbone. The remaining nodes serve as entry points for traffic from/to AT&T clients and are called *Points of Presence (POPs)*. This reflects the current EasyWWW architecture.

The sizes of the various documents were obtained from the trace, and bandwidth consumption was computed by multiplying the document size by the hops it traversed. Response latency was calculated as the sum of link latencies on the request and response paths, the document transfer time and the delay at the proxy server(s) and, if needed, the content server. The latency and bandwidth of various links in the topology were determined using delay measurement for various packet sizes. The proxy delays were obtained by assuming that they had a FIFO queue, with a fixed service time for each request. This model for latency on the backbone is not as realistic as for our bandwidth results, but it suffices to provide an idea of the penalty.

In our simulation, client requests are directed to the proxy server that is co-located with the exit point of traffic from Worldnet backbone to the client. Determining this exit point is easy for AT&T clients since each client is connected to a unique Worldnet node. For external clients, the exit point is determined by using actual Worldnet BGP tables; inconclusive BGP data are resolved using geographical proximity information from the Merit database [10]<sup>4</sup>. Selecting a reverse proxy based on BGP tables is used by many existing DNS load balancers, such as Distributed Director [5].

Since every AT&T client connects at a unique POP, proxies did not forward requests from these clients. As already mentioned, the response from a remote proxy would have come back to the client POP anyway, and so request forwarding would add latency without any bandwidth savings. For requests from outside clients, we deployed the forwarding mechanism at all four IGRs/proxies, assuming each of them had infinite cache capacity. Since total size of all documents is less than 3 GB, our assumption about cache capacity is reasonable for this trace. Section 4.2 assumes that each cache has perfect, instant knowledge of the content of remote caches. Section 4.3 then explicitly studies the effect of delayed and imprecise knowledge on the performance of our schemes.

We investigated the two configurations described in the preceding section. The first configuration, which we call *IGR*, forwards among reverse proxies. The second, which we call *Server-Forwardable*, also can forward requests to the server or web hosting complex

<sup>4</sup>more current geographical information is provided by Net-Geo servers [11].

and have the response sent back directly to the client.

A proxy needs to know whether to forward a request or fetch and cache the page. To do this end, we use a forwarding threshold. Each proxy maintains a reference counter for each document, and forwards the requests when the corresponding document’s reference counter does not exceed the threshold. Otherwise, the proxy fetches and stores the object. When there are multiple cached copies, a proxy picks the nearest remote proxy for forwarding or fetching.

In order to reflect the dynamics of documents, we introduce a new parameter called documents’ *time-to-live*. Our assumption here is that the content of the documents will change periodically. Old cache copies will be out-of-date, and should be discarded. The time-to-live give the interval between consecutive changes. We assume the time-to-live parameter is the same for all documents.

## 4.2 Performance

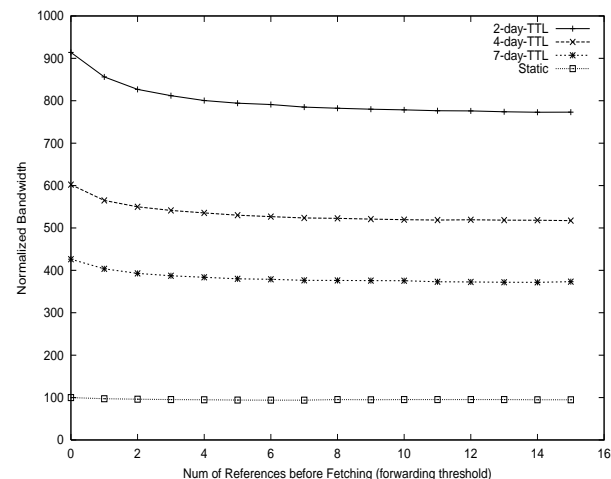


Figure 6: Bandwidth Under Different Document TTL (IGR)

Figure 6 and Figure 7 plot the total backbone bandwidth consumption of the trace for *IGR* and *Server-Forwardable* respectively. The curves in the graph show the bandwidth consumption under different values of the time-to-live parameter; “Static” represents the assumption that all the documents will never change; accordingly, “7-day-TTL” means the documents change every week. From the graphs, we can see that if all the documents are “static”, we can get up to 6% bandwidth reduction by for *IGR* policy and 20% for *Server-Forwardable*. This implies that if we tolerate several cache misses without fetching the object, we could filter out those seldom accessed

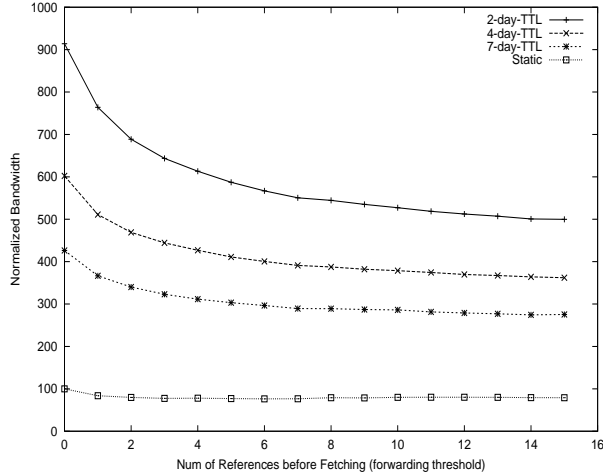


Figure 7: Bandwidth Under Different Document TTL (Server-Forwardable)

pages and only fetch those repeatedly accessed “hot” pages. Bandwidth savings increase significantly for shorter time-to-live values. In the 7-day TTL case, we can save up to 13% and 35% of bandwidth; in the 4-day TTL and 2-day TTL cases, the bandwidth saving can be more than 15% and 40%, for *IGR* and *Server-forwardable* respectively. All curves (almost) stabilize quickly, meaning a relatively small threshold suffices to get most of the benefit.

There are two reasons why bandwidth savings in *IGR* are not as large as that in *Server-Forwardable*:

1. For *IGR*, our technique starts to work when there is a miss on one proxy while the same request is a hit on another, but it seems that the chance of this situation is low for reverse proxies. Since we assume infinite cache capacity, especially in the case of static documents, once we cached the object, all the future references would be hits; there would not be many local misses we could forward. Considering document time-to-live parameters partly alleviates this problem by making document caches out-of-date and invalid, thus increasing local misses.
2. During the warming-up phase, when none of the IGRs has a copy of a document, even a reference to a rarely accessed document would lead to an expensive fetching operation, because there is no place to forward. This could also reduce the bandwidth margin we can save. In the *Server-Forwardable* case, however, any request will be a hit on the server. Therefore making the server capable of accepting forwarded requests will make forwarding much easier and thus fur-

ther reduce the bandwidth consumption.

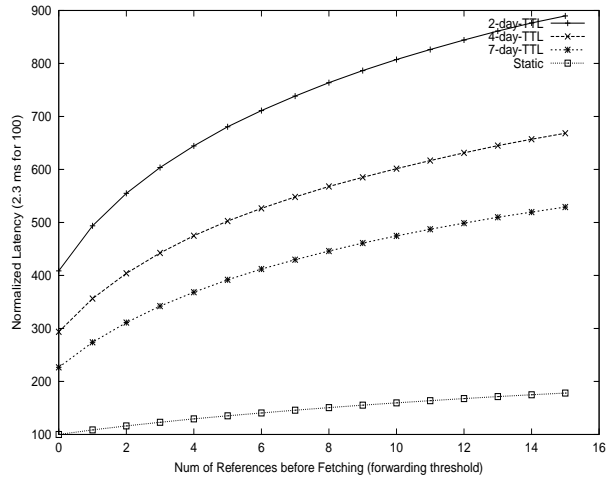


Figure 8: Latency Under Different Document TTL (*IGR*)

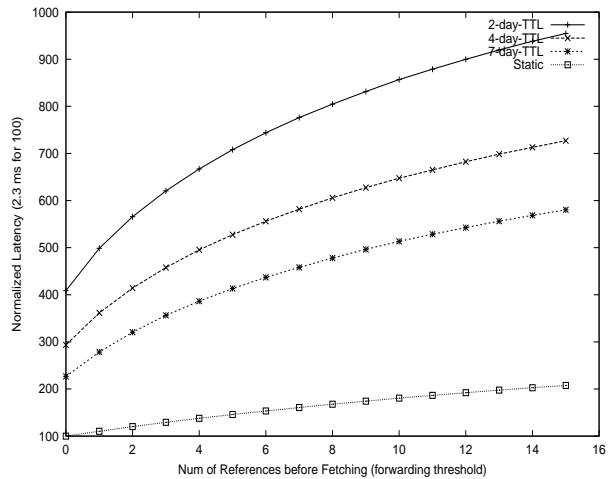


Figure 9: Latency Under Different Document TTL (*Server-Forwardable*)

Figure 8 and Figure 9 show the latency overhead brought by the forwarding mechanism. At first look, this is really a bad penalty: the latency roughly doubled for each case. However, these numbers only represent the latency on the backbone, which has a base overhead of 2.3 ms/req. Taking the 7-day-TTL case in *IGR* for an example, the worst latency on the backbone is 11 ms/req. How badly it affects the overall client-perceived latency depends on the speed of the client side link, but in the current Internet such delay overhead would be in the noise.

### 4.3 Delayed Dissemination of Cache Content Information

As mentioned earlier, we assumed so far that all proxies have instant, perfect knowledge of all cache locations of every requested document. We now consider how propagation delay of cache location information affects our results. To this end, we approximated the summary cache cooperation model [6]. In the simulations of this section, each proxy maintains a summary of other proxy’s cache information. All proxies exchange this information at designated time points. In other words, at the designated time points, all reverse proxies get up-to-date information; this information then becomes stale and imprecise due to changes in each proxy cache content, until next exchanging time point. On a false hit i.e., when a proxy contacts a remote proxy that does not have the document, this proxy will get a negative acknowledgement from the remote one and then fetch the document from the content server itself.

Figure 10 shows the delay impacts on bandwidth consumption for the *IGR* scheme. There is a gap between the curves for the cases with propagation delays and the curve for the perfect knowledge case. We found that the gap mostly comes from incomplete location information, causing a forwarding proxy to pick sub-optimal remote proxies in some cases. False hits are pretty rare and only account for less than 0.06% of total requests on any proxy. Overall, our technique shows similar trends in bandwidth savings with delayed or with instant dissemination of cache location information. While the delays do affect the amount of bandwidth savings, the performance gap between perfect and imprecise knowledge is rather small: for bandwidth, the gap is up to 5% of total traffic, and for latency, the absolute difference is less than 1.5ms.

Figure 11 shows the impact on latency in *IGR*. It shows that adding delays reduces latency overhead, compared to perfect knowledge. The reason for lower latency overhead is as follows: due to the propagation delay, a proxy receiving a request sometimes does not know that the requested object is cached on another proxy. The first proxy would then fetch the document from the content server rather than forward the request to the remote proxy. Thus, request forwarding is applied less frequently; therefore the latency overhead of request forwarding is reduced, and the reduction is accompanied by the corresponding reduction in bandwidth savings, as seen on Figure 10.

As for *Server-Forwardable*, where content server joins this forwarding mechanism, from Figure 12 and Figure 13, we see practically no difference between

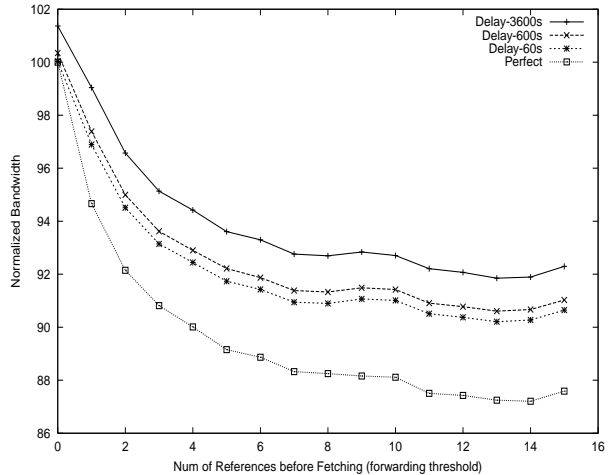


Figure 10: Impact of propagation delay of cache location information on bandwidth (*IGR*, 7-day-TTL).

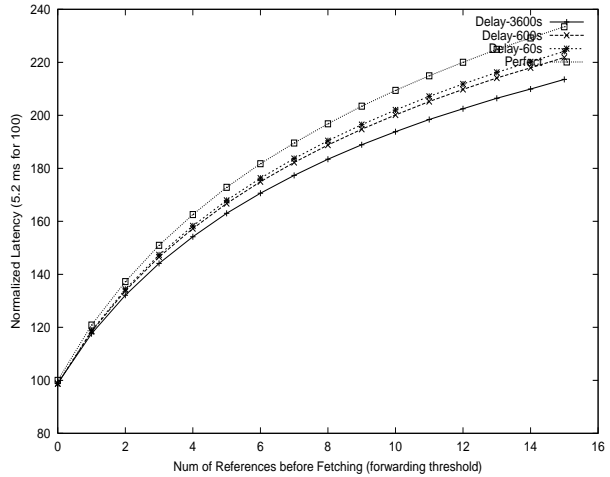


Figure 11: Impact of propagation delay of cache location information on latency (*IGR*, 7-day-TTL).

different cases. So, our mechanism is robust against information propagation delays when the server accepts forwarded requests.

## 5 Related Work

Several products (e.g., Network Dispatcher [7], Distributed Director[5], Radware [13], InterNAP [8], and ArrowPoint [2]) implement triangular communication at the switch and router levels. Also, LARD [12] deploys a similar TCP hand-off mechanism for request distribution in a clustered Web server. In this case, client requests arrive at a switch that forwards them to a server in a server farm. In contrast to these

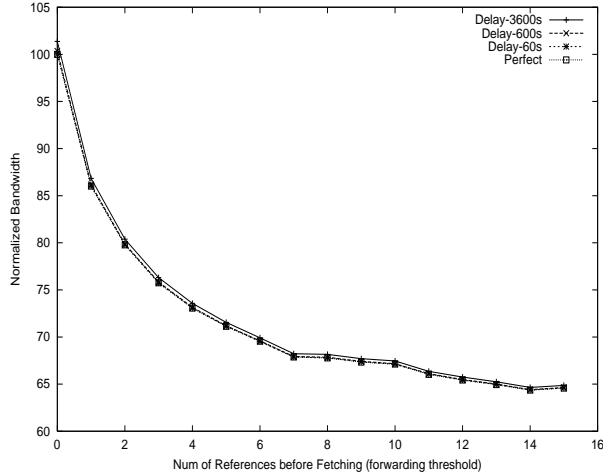


Figure 12: Impact of propagation delay of cache location information on bandwidth (*Server-Forwardable*, 7-day-TTL).

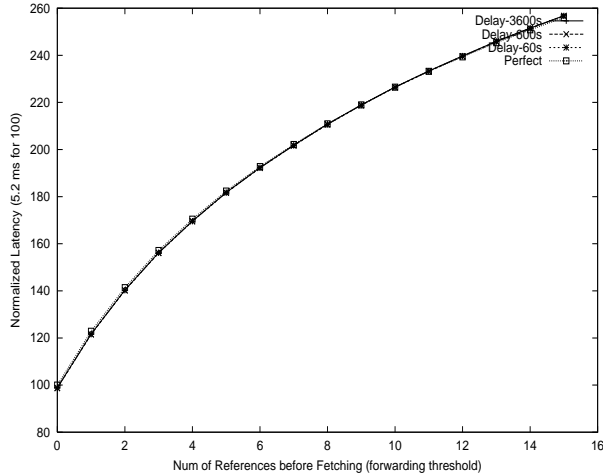


Figure 13: Impact of propagation delay of cache location information on latency (*Server-Forwardable*, 7-day-TTL).

products or research projects, we envision the triangular communication between Web proxies. This entails several differences between the two approaches. Among them, the need for the first proxy to complete the establishment of TCP connection and interpret the request (rather than blindly forward all packets); the need to interface with a cache cooperation mechanism to find a remote proxy with the requested page and the need for dynamic impersonation of the forwarding proxies (in the case of a switch, all servers are statically configured to always impersonate the IP of the switch; in our case, the second proxy must impersonate different proxies for differ-

ent requests). In this respect, our work looks more similar to Distributed Packet Rewriting (DPR) [3]. However, DPR is still targeted at relatively tightly coupled web servers. Our approach is deployed in a WAN environment.

Dynamic impersonation of other senders is also performed by transparent proxies. In this context, a switch or a router intercepts TCP port 80 traffic and diverts it to a proxy, which establishes the TCP connection with the clients impersonating the various Web servers they are trying to access. Technically, transparent proxies and switch-level triangular communication together implement most of the transport-level functionality required by our approach, giving us optimism in the feasibility of our proposal. At a higher level, our approach also requires interfacing with the proxy cooperation mechanism and the implementation of the synchronization protocol to fulfil the requirements of HTTP pipelining (see Section 3.1).

## 6 Conclusions and Future Work

In this paper, we have proposed using TCP connection forwarding among reverse proxies. This new technique for reverse proxy cooperation has several potential benefits: reducing bandwidth consumption on ISP’s backbone, balancing load among the proxies, better utilizing of client-proxy network proximity, and bypassing congested client-proxy connections.

Focusing on the issue of backbone bandwidth consumption, our simulation study demonstrated that connection forwarding results in noticeable bandwidth reduction. The reduction was 13% and 35% under the seven-day TTL assumption, depending on the policy used. The latency penalty was on the order of 10 milliseconds. Our experiments also showed that maintaining a dedicated output link at the content servers will greatly facilitate our mechanism, by improving the bandwidth savings from 13 to 35%, and at the same time make our approach more robust against different mechanisms for location information management.

We view our work presented in this paper a starting point for deploying forwarding requests mechanism within the context of cooperative proxies. We have already showed that this mechanism is useful in reducing web caching related backbone bandwidth consumption. In the future, we would like to investigate other potential benefits and to simulate the schemes on more extensive datasets. We also plan to refine our forwarding policy and implement a proto-

type system using Linux and Squid.

## Acknowledgments

We are grateful to Balachander Krishnamurthy at AT&T for providing us with the EasyWWW trace. We also benefitted from Euthimios Panagos's event simulator. Last, we thank all the people at AT&T Labs Research with whom we had interesting discussions.

## References

- [1] Mohit Aron, Peter Druschel, and Willy Zwaenepoel. Efficient support for p-http in cluster-based web servers. In *Proceedings of the USENIX 1999 Annual Technical Conference*, June 1999.
- [2] ArrowPoint Communications. <http://www.arrowpoint.com>.
- [3] A. Bestavros, M. Crovella, J. Liu, and D. Martin. Distributed packet rewriting and its application to scalable server architectures. In *Proceedings of ICNP'98: the 6th International Conference on Network Protocols*, Austin, Texas, October 1998.
- [4] Anawat Chankhunthod, Peter Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical internet object cache. In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996.
- [5] Cisco Systems, Inc. DistributedDirector. White paper. [http://www.cisco.com/warp/public/734/distdir/dd\\_wp.htm](http://www.cisco.com/warp/public/734/distdir/dd_wp.htm).
- [6] Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proceedings of the ACM SIGCOMM'98 conference*, pages 254–265, September 1998. <http://www.cs.wisc.edu/~cao/papers/summarycache.html>.
- [7] IBM Interactive Network Dispatcher. <http://www.ics.raleigh.ibm.com/netdispatch/>.
- [8] Internap network services. <http://www.internap.com/how.htm>.
- [9] David Maltz and Pravin Bhagwat. Application layer proxy performance using tcp splice. *Journal of High Speed Networks '99*, 1999.
- [10] Merit database. <ftp://nic.merit.edu/nsfnet/announced.networks/nets.unl.950428>.
- [11] NetGeo – the Internet Geographic Database. <http://www.caida.org/Tools/NetGeo/>.
- [12] Vivek Pai, Mohit Aron, Gaurav Banga, Michael Svenden, Peter Druschel, and Willy Zwaenepoel and Erich Nahum. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the 8th ACM Conference on Architectural Support for*
- Programming Languages and Operating Systems*, Oct 1998.
- [13] Radware. <http://www.radware.com/>.
- [14] K. Thompson, G. J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Networking*, 11(6), November 1997.