Evaluating Cooperative Web Caching Protocols for Emerging Network Technologies[1]

Christoph Lindemann and Oliver P. Waldhorst
University of Dortmund
Department of Computer Science
August-Schmidt-Str. 12
44227 Dortmund, Germany
http://www4.cs.uni-dortmund.de/~Lindemann/

Abstract

While bandwidth for previous IP backbone networks deployed by Internet Service Providers typically has been limited to 34 Mbps, current and future IP networks provide bandwidth ranging from 155 Mbps to 2.4 Gbps. Thus, it is important to investigate the impact of emerging network technologies on the performance of cooperative Web caching protocols. In this paper, we present a comprehensive performance study of four cooperative Web caching protocols. We consider the Internet cache protocol ICP, Cache Digests, the cache array routing protocol, CARP, and the Web cache coordination protocol, WCCP. The performance of these protocols is evaluated using trace-driven simulation with measured Web traffic from NLANR. The goal of this performance study lies in understanding the behavior and limiting factors of the considered protocols for cooperative Web caching under measured traffic conditions. Based on this understanding, we give recommendations for Internet Service Providers, Web clients, and Application Service Providers.

Key words:
Client, proxy and server-side caches of Web content,
Data and service replication on the web,
End-to-end performance evaluation of caching and replication,
Trace-driven simulation.

1 Introduction

Cooperative Web caching means the sharing and coordination of cached Web documents among multiple communicating proxy caches in an IP backbone network. Cooperative caching of data has its roots in distributed file and virtual memory systems in a high-speed local area network environment. Cooperative caching has shown to substantially reduce latencies in such distributed computing systems because network transfer time is much smaller than disk access time to serve a miss. While bandwidth for previous IP backbone networks deployed by Internet Service Providers typically has been limited to 34 Mbps, current and future IP networks provide bandwidth ranging from 155 Mbps to 2.4 Gbps (see e.g., CA*Net-3 [1], or Internet-2 [5]). Thus, it is important to investigate the impact of emerging network technologies on the performance of cooperative Web caching protocols.

Protocols for cooperative Web caching can be categorized as message-based, directory-based, hash-based, or router-based [4]. A popular example for a message-based protocol is the Internet cache protocol, ICP [10]. Directory-based protocols include Cache Digests [9]. The most notable hash-based cooperative Web caching protocol constitutes the cache array routing protocol, CARP [8]. An example for a router-based protocol is the Web cache coordination protocol, WCCP [3]. In previous work, the performance of cooperative Web caching protocols has mostly been studied just in comparison to ICP [7], [8], [9], [11]. To best of our knowledge, a comprehensive performance study of the protocols ICP, CARP, Cache Digests, and WCCP based on the same IP backbone network topology and workload characteristics has not been reported so far. Furthermore, the effect of rapidly increasing bandwidth availability to these cooperative Web caching protocols has not been investigated.

We present a comprehensive performance study for the cooperative Web caching protocols ICP, Cache Digests, CARP, and WCCP. The performance of the considered protocols is evaluated using measured Web traffic from NLANR [6] using a discrete-event simulator of an IP backbone network. The presented curves illustrate that for Internet Service Providers (ISPs) operating IP networks with high bandwidth

[1] This work was supported by the DFN-Verein with funds of the BMBF.

availability (622 Mbps and 2.4 Gbps), clearly either CARP or WCCP is the protocol of choice. From the clients' point of view, ICP is most beneficial because ICP achieves lowest latency. For Application Service Providers (ASPs), WCCP is the protocol of choice because WCCP achieves the best cost/performance ratio with respect to cache utilization. Sensitivity studies show that temporal locality and low cache utilization due to low participation to cooperative Web caching make considerably impact to the performance of individual protocols.

The paper is organized as follows. A brief description of the considered protocols is provided in Section 2. The simulation environment for evaluating the considered protocols for cooperative Web caching is introduced in Section 3. In Sections 4, we present performance curves for the considered protocols for cooperative Web caching derived from the trace driven simulator. Finally, concluding remarks are given.

2 Protocols for Cooperative Web Caching

The aim of Web proxy caching lies in reducing both document retrieval latency and network traffic. Cooperative Web caching means that if a requested document is not contained in the queried Web cache, other caches are queried first before the document is downloaded from the origin Web server. Protocols for cooperative Web caching can be categorized as message-based protocols, directory-based protocols, hash-based protocols, and router-based protocols. Message-based protocols define a query/response dialog for exchanging information about cached content. An example for a message-based protocol is the Internet cache protocol, ICP. Directory based protocols summarize information into frequently exchanged directories. Cache Digests constitutes an example for a directory-based protocol. Hash-based protocols as e.g., the cache array routing protocol, CARP, employ hash functions to distribute the URL space among cooperating Web caches. Router-based protocols intercept Web traffic on IP layer and redirect it to a caching proxy. An example for a router-based protocol constitutes the Web cache coordination protocol, WCCP.

In this section, we recall the major functionality of these protocols for cooperative Web caching in a mesh of loosely coupled Web proxy caches. Most protocols include further functionality, e.g., for dynamic reconfiguration of the caches or fault tolerance. This functionality is beyond the scope of our study and is omitted in the description. We describe the actions performed by a cache after a miss for a client request. We call the queried Web cache the *target cache*, all other caches in the mesh are denoted as *peer caches* [4].

ICP is an application layer protocol based on the user datagram protocol/internet protocol, UPD/IP. It was originally introduced by the Harvest Web cache for coordinating hierarchical Web caching. ICP defines a set of lightweight messages used to exchange information about cached documents among caches. If a target cache suffers a miss on a client request, it queries its configured peers using a special *ICP query* message. A peer cache replies with an *ICP hit* message, if it holds the requested document. Otherwise, the peer cache sends an *ICP miss* message. The target cache waits either an ICP hit is received or all configured siblings have reported a miss. Since ICP is based on UDP, messages are not reliably delivered. A timeout prevents a cache from infinite waiting for lost messages. The target cache fetches the document from the first peer cache signaling a hit. If no peer cache signals a hit, the document is fetched from the origin Web server or a configured parent cache. The document is stored locally at the target cache and delivered to the client. Note that with ICP multiple copies of a particular document may be stored in a cache mesh. ICP can adapt to network congestion and cache utilization by locating the peer cache which sends the fastest response. The disadvantage of ICP lies in introducing additional latency on a cache miss due to waiting until receiving replies from all peer caches or reaching the timeout.

Cache Digests defines a protocol based on the hypertext transfer protocol, HTTP. As ICP, Cache Digests operates on the application layer of the TCP/IP protocol stack. Cache Digests was introduced by Rousskov and Wessels and is implemented in the Squid Web cache [9]. Cache Digests enables caching proxies to exchange information about cached content in compact form. A summary of documents stored by a particular caching proxy is coded by an array of bits; i.e., so called *Bloom filter*. To add a document to a Bloom filter, a number of hash functions are calculated on the document URL. The results of the hash functions specify which bits of the bit array have to be turned on. On a lookup for a particular document, the same hash functions are calculated and the specified bits are checked. If all bits are turned on, the document is considered to be in the cache. With a certain probability, a document is erroneously reported to be in the cache. The probability depends on the size of the Bloom filter. Cooperating caching proxies build Cache Digests as a summary of locally cached content. Each target cache requests digests from all peer caches on a regular basis. If a target cache misses on a client request, it searches for the requested document in the digests of its peer caches. If the requested document is reported to be cached by any digest, the document is requested form the corresponding peer cache. Otherwise, the document is fetched from the origin Web server. In both cases, the target

cache stores a copy of the requested document and the document is delivered to the client. Note that as in ICP, multiple copies of a particular document may be stored in a cache mesh.

An advantage of Cache Digests lies in only generating network traffic on the digest exchange and not on each document request. The disadvantage of Cache Digests lies in causing *false cache hits*. False hits can occur in two ways: First, a digest reports that a particular document is cached, but the document has been already evicted since the generation of the digest. Second, a bloom filter may report a hit when the document is not in the cache. The number of false hits influences the performance of a cache mesh by causing unnecessary network traffic [9].

CARP constitutes also a cooperative Web caching protocol on the application layer of the TCP/IP protocol stack. CARP is based on HTTP and was introduced by Valloppillil and Ross [8]. As Cache Digests, CARP is based on hash functions. The version of CARP we consider in Section 4 implements hashing inside the caches. Thus, the considered version of CARP does not require changing the client software. Note that another version of CARP is implemented such that hashing is performed in the client software [8]. Each cache in the cache mesh stores an unique hash function, which maps the URL space to a hash space. Each element of the hash space is associated with a particular caching proxy. If a target cache receives a client request, it calculates the hash key of the document URL. The result is combined with the hash keys of the names of all caches in the mesh. The request is forwarded to the cache, which achieves the smallest value of the combined hash key. We refer to this cache as the *selected cache*.

The selected cache can be any member of the cache mesh including the target cache. On a hit, the selected cache delivers the document to the target cache, if both are different. On a miss, the selected cache retrieves the document from the origin Web server, stores a copy and delivers it to the target cache. The target cache delivers the document to the client. Note that the target cache does not store a copy of the document. Each document is stored only at the selected cache. This unifies several physically distributed caching proxies to one logical cache. The advantage of CARP lies in not wasting cache memory for replicated documents. As a disadvantage, CARP employs expensive HTTP communication among the caching proxies on each request.

WCCP is a router-based protocol on the network layer of the TCP/IP protocol stack. Recently, the specification of WCCP v2.0 has been made public available in an Internet draft [3]. WCCP takes the responsibility for the distribution of requests away from the caches. It allows a Web cache to join a *service group*. A service group consists of one or more caches and one ore more routers. The routers of a service group intercept IP packets of HTTP requests sent from a client to the origin Web server. The packets are redirected to a particular caching proxy in the service group. As in CARP, a hash key chooses the caching proxy. Opposed to CARP, in general the router does not know the complete URL of a requested document. Therefore, it uses the IP address of the destination of intercepted packets as input for the hash function, i.e., the IP address of the origin Web server holding the requested document. The result of the hash function yields an index into a redirection hash table. This redirection hash table provides the IP address of the Web cache to which the request is forwarded. Due to hashing based on IP addresses, all objects of a particular site are serviced by the same caching proxy. This constitutes the main difference to the version of CARP we consider. The selected cache services the request from cache storage or the origin Web server. The requested document is delivered directly to the client by the cache. Note that as in CARP, at most a single copy of a document is stored in a local cache mesh. As a further advantage, WCCP does not increase the load on caching proxies by distributing documents via the routers. As a disadvantage, WCCP increases the load on routers due to the calculation of hash functions and packet redirection.

As mentioned above, we use implementations that do not require changes in user agents. That is the client has to be configured to issue document request to a statically assigned caching proxy or the origin Web server.

3 Simulation Environment

3.1 Network model

In our experiments, we investigate the performance of cooperative caching for caching proxies distributed across the backbone network of a national Internet service provider (ISP). We assume that caches are placed at five access routers, which connect one or more regional networks to the backbone. Clients of a national cache consist of user agents, institutional and regional caches, which are directly connected to the associated access router via institutional and regional networks. Our simulator comprises of several building blocks, which can be utilized to assemble arbitrary configurations for backbone networks. These building blocks are CSIM simulation components modeling virtual links, access routers, clients, caches, and Web servers. Figure 1 illustrates the network configuration used for the performance studies presented in Section 4. All parameters of the network model are shown in Table 1.

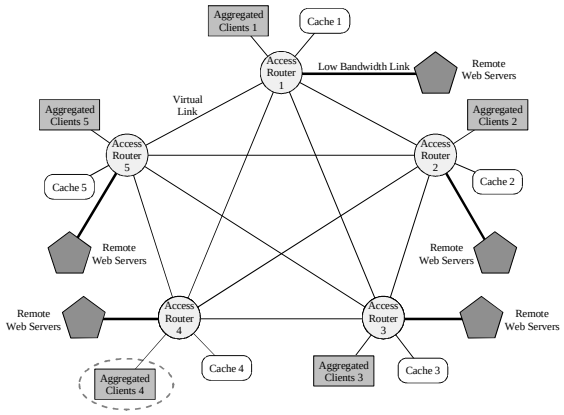

Figure 1. Model of network configuration with virtual links and aggregated clients

Virtual links

The simulator represents connectivity of basic components by virtual links. That is as simplifying assumption; we do not represent each physical link in the local national backbone network. A virtual link connects each pair of access routers. Each virtual link is associated with a round trip time RTT, i.e., the time, which is needed by an IP packet to be transmitted from the source router of the virtual link to the destination and back. Furthermore, a packet loss probability P_{loss} is assigned to each link. This is the probability that a packet is lost while being transmitted on a virtual link. Virtual links are used to transmit two types of messages. The first type are ICP messages which use UDP/IP as underlying transport protocol. We assume an ICP message to get lost with probability P_{loss} and to get delayed with a mean of $RTT/2$ otherwise. The second type of transmission is a HTTP request/response dialog. HTTP uses the transmission control protocol/internet protocol, TCP/IP as underlying transport protocol. To capture the mean delay introduced by TCP/IP, we employ the analytical model introduced in [2]. This model takes RTT and P_{loss} as well as some parameters of the TCP/IP protocol stack together with the size of the transmitted message as input for deriving the mean delay for connection setup and data transmission. To model the request part of HTTP, we calculate the mean time for connection set up as well as the transmission time for the request in TCP's slow start mode. For the response, we assume the mean delay for document transmission consisting of the slow start phase and a following steady state transmission phase if the document is large enough. We make the assumption that RTT and P_{loss} are not significantly influenced by traffic originating from inter-cache communication, i.e., they are constant during the simulation run. This assumption holds for inter-cache traffic in most national backbone networks because of low cache utilization.

Parameter	Description	Value
RTT	Average round trip time in local backbone network	varying
P_{loss}	Packet loss probability in local backbone network	0.01 %
T_{HTTP}	Time for of HTTP connection	3.0 msec.
T_{Cache}	Time for cache lookups/updates $(0.5*T_{HTTP})$	1.5 msec.
T_{ICP}	Time for processing an ICP query/response $(0.1*T_{HTTP})$	0.3 msec.
T_{CARP}	Time for calculation of destination cache $(0.2*T_{HTTP})$	0.6 msec.
T_{WCCP}	Time for calculation of target IP address and processing encapsulated IP packets $(0.1*T_{HTTP})$	0.3 msec.

Table 1. Parameters of the simulator

To evaluate cooperative Web caching for emerging network technologies, we need to configure virtual links for different bandwidth available in the local backbone network. In our experiments, we examined network bandwidth of 34 Mbps, 155 Mbps (STM-1 connection), 622 Mbps (STM-4 connection), and 2.4 Gbps (STM-16 connection). We estimate round trip times for links offering different bandwidth. For a 155 Mbps and 622 Mbps bandwidth, we assume an average round trip times of 15 msec. and 10 msec., respectively. These delays are observed in current Gigabit networks. To estimate round trip times for 34 Mbps and 2.4 Gbps connections, we employ linear regression. From the measured delays, we observe that increasing network bandwidth by factor four approximately decreases average round trip time by 5 msec. Thus, we assume average round trip times of 20 msec. and 5 msec. for 34 Mbps and 2.4 Gbps connections, respectively.

Local Web Caches

Web proxy caches are connected to the access routers connecting regional networks to the local national backbone. The simulator implements the basic functionality of a caching proxy similar to that described in [12]. Each cache implements a cache manager using the LRU replacement scheme and a CPU server servicing requests in a FIFO queue. The cache manager can be configured to

provide the functionality of ICP, Cache Digests, or CARP as described in Section 2. The core functionality of WCCP is implemented in the routers and, therefore, is not part of the cache manager. The simulator does not represent modifications of Web documents. Thus, the cache manager does not expire documents as real Web caches do. Therefore, the simulator slightly overestimates the amount of traffic saved by cooperative Web caching. The service time of the CPU server depends on the operation performed. For processing a HTTP request time T_{HTTP} is required. Time for cache lookups and updates is denoted by T_{Cache}. Following [12], we assume that $T_{Cache} = 0.5 \cdot T_{HTTP}$. A single ICP query or response is processed in time T_{ICP}. It holds $T_{ICP} < T_{HTTP}$, since ICP messages are short and simple UDP transmissions [10]. Therefore, we assume that $T_{ICP} = 0.1 \cdot T_{HTTP}$. Calculation of hash functions takes time T_{CARP} for CARP. The hash function defined by CARP uses several arithmetic operations as 32 bit multiplication [8]. Therefore, we assume that hashing is more expensive than processing ICP messages, i.e., $T_{CARP} = 0.2 \cdot T_{HTTP}$. We assume that calculation of an entry of a cache digests based on hash functions also takes the delay T_{CARP}. Requesting and sending digests is performed based by HTTP and takes the delay T_{HTTP}.

The delay T_{HTTP} is determined experimentally as follows. Following [12], we perform simulation runs for various values of T_{HTTP} and monitor the utilization of the simulated caches for each value of T_{HTTP}. Since the correct cache utilization has also been measured during trace recording, the values for cache utilization corresponding to different values of T_{HTTP} are used to calibrate the delay T_{HTTP}. This calibration yields T_{HTTP} = 3.0 msec.

Access routers

We refer to a router connecting one or more regional networks to the national backbone network as an *access router*. Access routers receive transmissions from a client, Web cache, or link component and pass it to an other Web cache, client, or link. Because all network transmission delays are included in the round trip time associated with virtual links, an access router does not further delay a transmission. The transmission redirected by WCCP constitutes an exception to this. In WCCP, an access router has to calculate the new target cache of a client request using a hash function and encapsulate the IP package for redirection. T_{WCCP} denotes the combined time needed for these operations. Opposed to CARP, the hash function defined by WCCP is based on simple bit shifting and XOR operations, which can be easily calculated by the router [3]. Redirection is done by rewriting a IP packet with an additional header, which is also a fast operation. Therefore, it holds $T_{WCCP} < T_{CARP}$, and we assume $T_{WCCP} = 0.1 \cdot T_{HTTP}$.

Clients

Several user agents, institutional, and regional caches can be connected to an access router via local and regional networks. The comparative simulation study presented in Section 4 focuses on distributed caching in a national backbone network. Thus, we refer to all these sources of requests as clients. We assume that all clients are configured to peer with the closest national cache, i.e., the cache located at the access router connecting their regional network to the national backbone network. We aggregate all clients connected to one access router in one clients component. The clients component assumed to be connected to the access router via a client link. Client links are used to transmit HTTP responses and introduce the same delay to a transmission as a virtual link does. The clients component issues requests to the caching proxy located at the same router. The request stream consists of the aggregated request streams of all clients located in the lower level networks connected to the central router. The request interarrival times are given by the timestamps of the requests recorded in the trace. A detailed description how client request streams are obtained from the trace files is given in Section 3.2.

Origin Web Servers in Remote Backbone Network

In the simulator, all origin Web servers are assumed to be located in a remote national backbone network. This remote network is connected via a low bandwidth connection to the local backbone network in which distributed Web caching using the considered protocols is explicitly represented. This low bandwidth connection to the remote backbone clearly dominates the delay for receiving a document from its origin Web server in the remote network. To reach the origin Web server, additional delays are introduced by passing the national, regional and institutional networks under the remote backbone network. However, these delays are negligible compared to the delay between two backbone networks. Finally, the utilization of the origin Web server constitutes another important factor for the overall document retrieval latency. The sum of these latencies is denoted as the download time of a Web document. Estimates for these mean delays are derived from the considered traces as described in Section 3.2.

3.2 Characteristics of the Considered Workload

To derive meaningful performance results for distributed Web caching in a national backbone network, we need to feed our simulator with representative workloads collected at top-level caches. The performance studies presented in Section 4 are based on traces from log files of ten top-level proxy caches of the National Laboratory for Applied Network Research (NLANR) cache hierarchy [6]. At time of trace collection 584 individual clients peered to the caches. We evaluated the trace files for a period of one week. Due to space limitations, we present only the results for a single day in the particular week. For all other days recorded in the trace files, similar performance results will be achieved.

To make the trace files usable for our performance study, some preprocessing has to be performed. First, from request recorded in the trace file we consider only GET requests to cacheable documents. We exclude uncachable documents by commonly known heuristics, e.g., by looking for string "cgi" or "?" in the requested URL. From the remaining requests, we consider responses with HTTP status codes 200 (OK), 203 (Non Authoritative Information), 206 (Partial Content), 300 (Multiple Choices), 301 (Moved Permanently), 302 (Found), and 304 (Not Modified) as cacheable. To determine the correct size of data transmissions with status code 304, we scanned the trace in a two pass scheme. Furthermore, we exclude all log file entries indicating communication between the caches, because the simulator all performs inter-cache communication. Trace characteristics after preprocessing are shown in Table 2.

Recall that the modeled network configuration contains five cooperating Web caches. Following [11], the overall request stream recorded in a measured trace is divided into fractional request streams issued by individual clients (i.e., 584 request streams for NLANR). Subsequently, these fractional request streams are agglomerated into five subtraces, one for each aggregated clients of Figure 1, such that the total number of requests issued by clients is approximately equal for each subtrace. Table 3 states the properties of the subtraces for NLANR.

To estimate the mean delays for retrieving documents from the origin Web server, we consider requests recorded with a TCP_MISS/200 status written by the Web proxy cache Squid. The transmission of a document from a cache to the client can be widely overlapped with the delay for transmitting the document form the origin Web server to the cache. Therefore, the time needed by a top level cache to retrieve a document from the origin Web server and deliver it to the client is a close approximation for the download delay specific for the document. If a document appears in several log file entries with status code TCP_MISS/200, we calculate download delay as mean value of the recorded time intervals. By scanning log files for an entire week, we were able to determine download delays for about 98% of the requested documents.

Trace	**NLANR**
Date	December 6, 2000
Duration (hours)	24
Number of Web documents	1,935,226
Overall document size (GB)	23.07
Number of requests	3,361,744
Requested data (GB)	64.11
Average requests/sec.	38.90
Number of clients	584

Table 2. Properties of traces measured in DFN and NLANR

The important factor for the performance of cooperative Web caching in the backbone of a national ISP is the ratio between the document transmission time in the local backbone network to the document download time from the origin Web server. Table 4 provides numerical results for the average download times derived from the NLANR traces as well as mean document transmission times for different network bandwidth. These mean document transmission times are derived by the performance modeling for TCP latency of [2]. From Table 4, we conclude that for a bandwidth of 622 Mbps in the local backbone network, the ratio between the document transmission time and the document download time is about 1/15 for NLANR.

4 Comparative Performance Study of Considered Protocols

4.1 Bandwidth Consumption

In a first experiment, we study bandwidth consumption in the cache mesh of the local backbone network for the considered cooperative Web caching protocols. Figure 2 plots the amount of saved traffic, i.e. the volume of requests served form inside the cache mesh, as a function of individual cache size for the NLANR trace. The results show that without cooperative Web caching an amount of 26 GB is saved, i.e., 39% of the overall requested data.

	NLANR
Average number of requests	672,349
Average requested data (GB)	12.82
Average requests/sec.	7.78

Table 3. Properties of DFN and NLANR subtraces for modeling aggregated clients

Figure 3 plots the inter-cache traffic as a function of cache size for the NLANR trace. Note that without cooperative Web caching no inter-cache communication occurs. For CARP and WCCP, we observe that the amount for inter-cache communication is about 80% of the requested total amount of data. ICP introduces overhead traffic for every cache miss in a target cache [9] and therefore generates medium cache traffic. Cache Digests cause the smallest amount of inter-cache communication for because Cache Digests does not introduce communication on every request as CARP or on every miss as ICP. We conclude form Figures 2 and 3 that hash-based schemes, as WCCP and CARP, are the protocols of choice for ISP if sufficient bandwidth is available inside the local backbone.

4.2 Document Retrieval Latency

We provide a comparison of the latencies introduced by the protocols for increasing cache sizes. The experiments are performed for bandwidths of 155 Mbps and 2.4 Gbps in the local backbone. Performance curves for the experiments are shown in Figures 4 and 5. We observe that Cache Digests shows worst performance of all protocols, while ICP yields lowest latency. We conclude that from the client's point of view, ICP is the protocol of choice. Comparing the results for bandwidths of 155 Mbps and 2.4 Gbps, we observe that the impact of bandwidth availability in the local backbone network on document retrieval latency is rather low. In order to further illustrate this effect, we keep in Figure 6 the relative cache size fixed to15% and investigate document retrieval latency for network bandwidth of 34 Mbps, 155 Mbps, 622 Mbps and 2.4 Gbps. We observe that document retrieval latency scales linear when bandwidth availability in the backbone network scales by factor four. From Figure 6, we conclude that the availability of additional bandwidth in the local backbone network has little impact on document retrieval latency. Currently, document retrieval latency is dominated by document download time observed in the traces. To show the impact of the ratio between document download time and document transmission time, Figure 7 presents a sensitivity study. From this study, we conclude that reducing download delays is a more effective approach for reducing client latency than cooperative web caching.

	Bandwidth	NLANR
Document transmission time in local backbone network	2.4 Gbps	224.21 msec.
	622 Mbps	243.72 msec.
	155 Mbps	263,28 msec.
	34 Mbps	282.74 msec.
Document download time	unknown	3633.88 msec.

Table 4. Document transmission and document download times derived from trace data

4.3 Cache Utilization

Figure 8 plots the average cache utilization as a function of cache size for the different protocols. Without cooperative Web caching low cache utilization is achieved because besides a constant number of client requests no further messages are generated. Opposed to this, ICP yields the highest utilization due to its overhead on a cache miss. WCCP offers lowest load to the caches due to balancing requests in the routers. From Figures 8, we also observe that the NLANR trace offers a moderate load to the caches. To investigate protocol performance for increasing request rates, we perform a sensitivity study shown in Figure 9. The relative cache size is kept fixed to 15%, network bandwidth is fixed to 622 Mbps and the request interarrival times in the traces are varied. We find that relative order in cache utilization between the protocols does not change with increasing workload. ICP achieves highest cache utilization because of its overhead on each miss. WCCP achieves even better scalability than no cooperative Web caching because of its load balancing among the caches. To ensure scalability, WCCP is clearly the protocol of choice.

Conclusion

We presented a comprehensive performance study of the cooperative Web caching protocols ICP, CARP, Cache Digests, and WCCP. We investigated performance of individual protocols from viewpoints of ISPs, clients, and Internet enterprises like ASPs. Consequently, as performance measures we considered bandwidth consumption, user latency, and cache utilization.

Recall that ISPs are most interested in the amount of saved traffic and in protocol efficiency. The curves presented in Figures 2 and 3 indicate that ISPs clearly benefit from cooperative Web caching. For currently operating backbone networks having

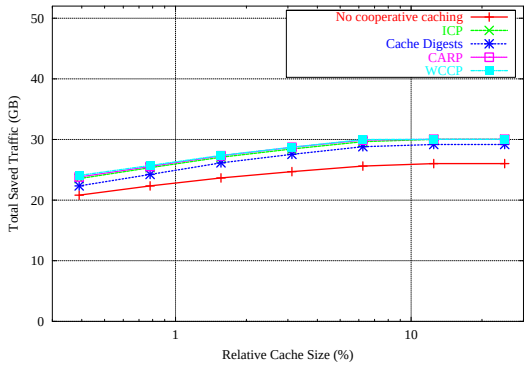

Figure 2. Bandwidth consumption of different protocols; saved traffic; 622 Mbps bandwidth

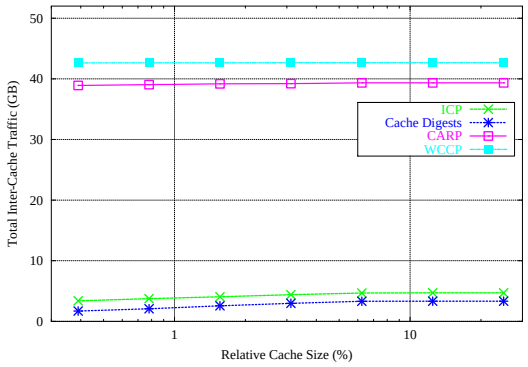

Figure 3. Bandwidth consumption of different protocols; inter-cache traffic; 622 Mbps bandw.

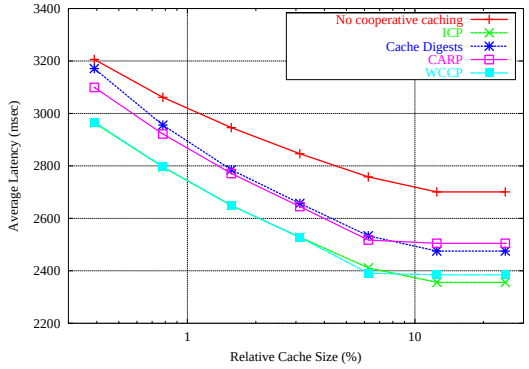

Figure 4. Document latency of different protocols; 155 Mbps bandwidth

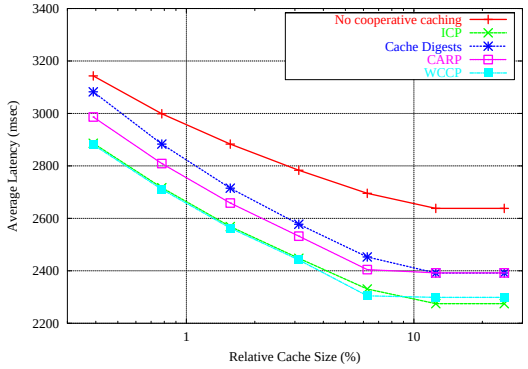

Figure 5. Document latency of different protocols; 2.4 Gbps bandwidth

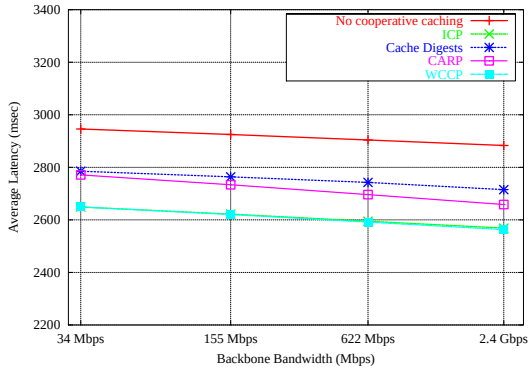

Figure 6. Document latency for different bandwidth availability

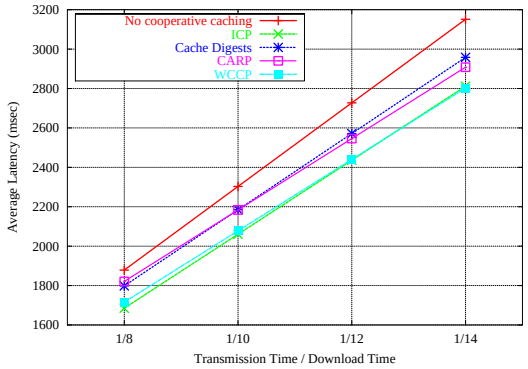

Figure 7. Sensitivity study: Latency vs. transmission/download ratio; 622 Mbps bandw.

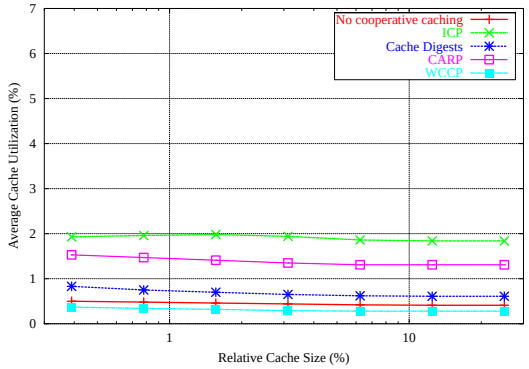

Figure 8. Cache utilization versus cache size; 622 Mbps bandwidth

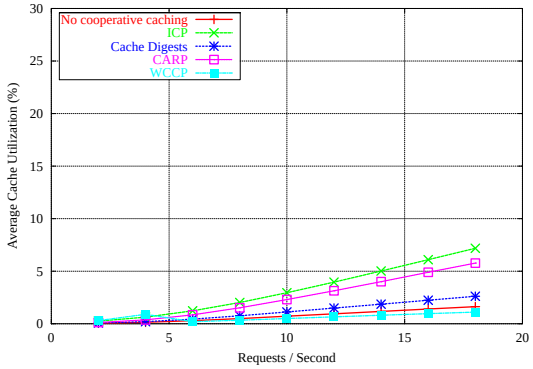

Figure 9. Sensitivity study: Cache utilization vs. request rate; 622 Mbps bandwidth

a bandwidth of 155 Mbps, ICP is the protocol of choice because of its high protocol efficiency. For future backbone networks having bandwidth of 2.4 Gbps, WCCP is superior to ICP because WCCP yields higher saved traffic than ICP while the low protocol efficiency of WCCP does not matter. Recall that ASPs are most interested in low cache utilization because this implies good cost/performance trade-offs. As illustrated in Figures 8 and 9, ASPs benefit from cooperative Web caching in case of future backbone networks providing bandwidth of 622 Mbps or 2.4 Gbps. In this case, WCCP yields the lowest cache utilization of all protocols and also lower cache utilization than no cooperative Web caching. Recall that clients are most interested in low document retrieval latency. We observe only little differences in document retrieval latencies achieved by the individual protocols. Moreover, these latency curves are in the neighborhood of the latency curve for no cooperative Web caching. Thus, clients have not much benefit from cooperative Web caching regardless which protocol is employed. As illustrated in a sensitivity study, clients will benefit most from reducing the ratio between transmission time and document download time. Therefore, providing higher bandwidth in links to remote backbone networks or deploying a content delivery network are considerably more effective approaches for reducing document retrieval latency than cooperative Web caching. Furthermore, we found that the impact of low cache utilization due to low participation in cooperative Web caching make considerably impact to the performance of individual protocols.

References

[1] CA*–Net 3, *Canada's Advanced Internet Development Organization*. http://www.canarie.ca/advnet/canet3.html

[2] N. Cardwell, S. Savage, and T. Anderson, Modeling TCP Latency, *Proc. 20th Conf. on Computer Communications (IEEE INFOCOM), Tel Aviv Israel*, 2000.

[3] M. Cieslak, D. Foster, G. Tiwana, and R. Wilson, Web Cache Coordination Protocol v2.0. *IETF Internet draft, 2000*.

[4] I. Cooper, I. Melve, and G. Tomlinson, Internet Web Replication and Caching Taxonomy, *IETF Internet draft, 2000*

[5] Internet–2, *University Corporation for Advanced Internet Development*. http://www.internet2.edu/

[6] NLANR, National Laboratory for Applied Network Research, http://www.nlanr.net/

[7] P. Rodriguez, C. Spanner, and E.W. Biersack, Analysis of Web Caching Architectures: Hierarchical and Distributed Caching, *IEEE/ACM Trans. on Networking 2001*

[8] K.W. Ross, Hash Routing for Collections of Shared Web Caches, *IEEE Network*, **11**, pp. 37-44, 1997.

[9] A. Rousskov and D. Wessels, Cache Digests, *Computer Networks and ISDN Systems*, **30**, pp. 22-23, 1998.

[10] D. Wessels and K.C. Claffy, ICP and the Squid Web Cache, *IEEE Journal on Select. Areas in Comm.*, **16**, pp. 345-357, 1998.

[11] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, On the Scale and Performance of Cooperative Web Proxy Caching, *Proc. 17th ACM Symp. on Operating Systems Principles (SOSP), Kiawah Island SC*, pp. 16-31, 1999.

[12] K.-L. Wu and P.S. Yu. Latency-sensitive Hashing for Collaborative Web Caching, *Proc. 9th World Wide Web Conference, Amsterdam The Netherlands*, pp. 633-644, 2000.