Acceleration of WWW Service with Distributed Cache Technology

Eiji Kawai

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

abstract:

Acceleration of WWW Service with Distributed Cache Technology

Great varieties of cache technologies are applied to improve the quality of WWW services. A distributed WWW cache system is such a strong solution where cache of a single proxy server is shared by other proxy servers. It can reduce WWW service latency, traffic on the Internet, load of WWW servers, etc. Internet cache protocol (ICP) is one of the protocols widely employed by the distributed WWW cache systems. Using ICP, the proxies scattered all over the world can easily cooperate with each other and increase their cache hit rate. On the other hand, the traffic generated by ICP might cause a consumption of network resources because it does not usually convey any WWW object data. Despite this disadvantage, ICP has been employed even in inappropriate situations because there is no quantitative analysis of the ICP traffic. Obviously, a system analysis based only on the hit rate, which is the most frequently used index for performance of a cache system, is inadequate to evaluate the effect of the distributed WWW cache system.

In this dissertation, firstly I give quantitative traffic analyses of distributed WWW cache systems especially focused on local area networks (LANs) and wide area networks (WANs). From a viewpoint of LAN, ICP simply increases the amount of ICP traffic passing through the boundary router that connects the LAN and the rest of the world. Because the network bandwidth between the LAN and the WAN is highly limited in many cases, ICP can cause a severe performance degradation. From a viewpoint of WAN, ICP can actually decrease the HTTP traffic on the Internet by retrieving HTTP objects not from distant origin servers but from possibly closer proxy servers. However, the ICP traffic on the Internet might cancel the traffic reduction. In my analysis, I figure out the communication model of the system and investigate the influence of several parameters on the traffic. The contribution of this dissertation includes that it specifies the proper and improper situation where ICP can be employed. With this analysis, WIDE cache bone, one of the largest distributed WWW cache systems in Japan operated by WIDE project, is reconfigured at many proxy servers to cease

sending ICP queries.

Next, I propose a new algorithm for a distributed cache system in place of ICP. ICP is a loosely coupled distributed cache system, i.e., cache nodes work independently of each other. On the other hand, tightly coupled systems where each cache node is assigned a portion of name space of WWW objects and caches only the objects whose names are in the assigned space. Hash routing is an algorithm for a tightly coupled cache system that achieves a high hit rate by preventing overlaps of objects between caches. One of the drawbacks of hash routing, however, is its weakness against failure. When one of the cache nodes fails, all the users using the system suffer from the performance degradation. In this dissertation we propose a *duplicated hash routing* algorithm that achieves high tolerance against the failure of cache nodes. Duplicated hash routing introduces minimum redundancy to keep system performance when some cache nodes are crashed. In addition, each node caches objects requested by its local clients (*local caching*), which may waste some portion of system cache capacity but it can cut down the network traffic between cache nodes. We evaluate various aspects of the system performance such as hit rates, error rates and network traffic by simulations and compare them with those of other algorithms. The results show that our algorithm achieves both high fault tolerance and high performance with low system overhead.

keywords: distributed WWW cache systems, Internet Cache protocol, traffic analysis, hop count, fault tolerance, hash routing, cache redundancy

World Wide Web

WWW
WWW

WWW
WWW WWW

(ICP) ICP WWW
Harvest Squid
ICP
ICP
WWW WWW
(HTTP)

ICP
ICP
ICP
(LAN) (WAN)
LAN ICP LAN

ICP
WAN ICP

HTTP
ICP
WWW

ICP

WWW WIDE cache bone
ICP ICP

ICP
ICP

WWW

( ) ICP

: WWW

Acknowledgements

I wish to express my gratitude to professor Kotaro Minato, my advisor and committee chairman, for his support, advice and encouragement. Without his kind cooperation, I could not complete this dissertation. I would also like to thank professor Akira Fukuda for his helpful support on my committee. His useful comments helped me to finish my work. My special thanks go to professor Suguru Yamaguchi for his continuous support and assistance with his great patience. He read my research papers carefully and gave me a lot of useful comments. I wish to thank associate professor Hideki Sunahara for his long sustained support to my work. His kind support encouraged me to make my research a fruitful one.

I also thank assistant professor Ken-ichi Chinen for exciting discussion. He also taught me a lot of programming techniques and I could implement various efficient simulation programs. I thank assistant professor Katsuyoshi Iida and my colleague Yutaka Nakamura for their friendly assistant and discussion that helped me to polish my research ideas. Their continuous stimulation push me way to the goal. Special thanks to Kadohito Osuga for his help to my research especially in the area of duplicated hash routing. I also thank all the members of Information Technology Center for their kind support and comments.

Finally I wish to thank my family. Their sincere encouragement and support enable me to go through my degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Although World Wide Web (WWW) was designed and developed originally as a medium for exchange of research information, its handy ways of publishing and retrieving various information have made its popularity fairly large. This enormous popularity of WWW, however, has raised various problems such as traffic congestion on back bone networks and exceedingly high load on WWW servers. Because all these problems increase service latency at clients, WWW is sometimes called cynically as *World Wide Wait*.

One of the popular techniques applied to resolve this problem is caching [1, 2]. Once a WWW object is stored in a cache, the object can be retrieved not from the origin server but from the cache. Thus the cache can reduce accesses to the origin server, network traffic on the Internet if the cache is near the clients, and also reduce the service latency experienced by the clients.

There are many points where the caching technology can be applied. First, each client (WWW browser) has its own cache. If the user revisits a WWW page, the objects composing the page can be retrieved from the cache. A proxy server is another point that can implement a cache. It is usually settled near the boundary between a local area network (LAN) and a wide area network (WAN) and relays the HTTP traffic crossing the boundary. Many organizations protect their LAN with a firewall that filters out any traffic other than that from/to permitted hosts and ports. In such case, aggregating the traffic from scattered clients at the proxy server makes the management of the firewall more simple and keeps the high security. These proxy servers with a cache are referred to as caching proxies. The caching proxy has an advantage that an object stored in the cache as a result of some client's request can be shared by other clients. On the other hand, a caching proxy server on the side of servers, a reverse proxy, also can alleviate some bottlenecks. The reverse proxy reduces

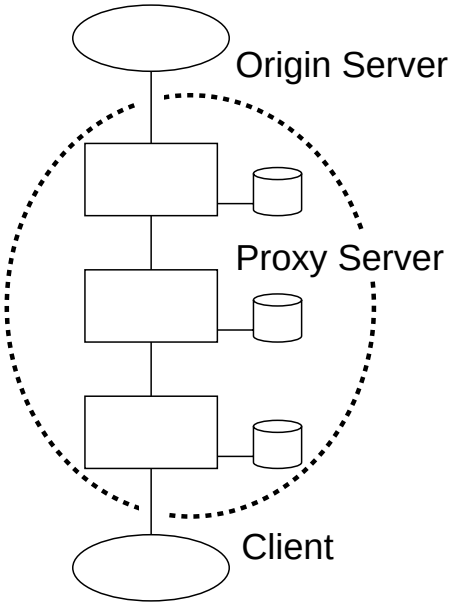

Figure 1.1: Systems connected in serial

the requests arriving at the servers, that is, the load of the servers.

In this dissertation, we focus on caching proxies, especially, distributed caching systems composed by scattered caching proxies on the Internet. With a distributed caching system, higher hit rate can be achieved than with a single caching proxy. The distributed caching systems can be classified into two categories, systems connected in serial and systems connected in parallel. Figure 1.1 and Figure 1.2 depict these systems.

In a system connected in serial, each caching proxy server receives HTTP requests and searches its local cache for the requested object. If the object is found in the cache, it is returned to the client. Otherwise, the request is forwarded upstream to the next proxy server, which is called a *parent*, or the origin server.

In a system connected in parallel, if a caching proxy server that receives an HTTP request does not have the requested object, it broadcasts queries to a set of neighbor proxy servers, which are called *siblings*. Each sibling returns the reply to the query and the proxy server can retrieve the object from one of the siblings that returned affirmation. If no affirmation is returned from the siblings, the proxy server forwards the HTTP request to the origin server.

In practical operation, a hybrid topology is usually employed. From a global viewpoint, each proxy server is serially connected and organizes a hierarchical tree structure. On the other hand, from a local viewpoint, some of them are interconnected in parallel with a relationship of siblings.

Not only a strict tree structure, but also a non-tree structure might also be employed

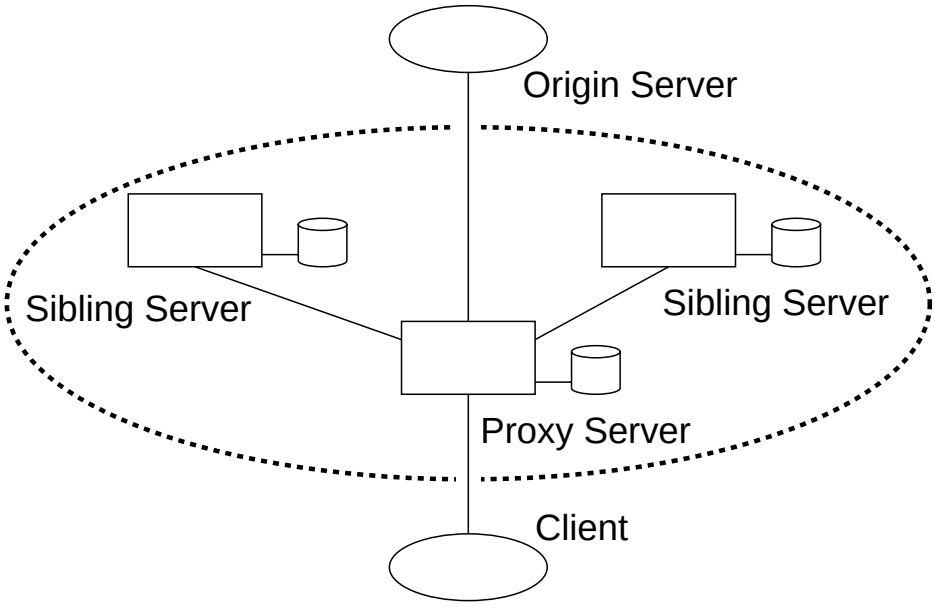

Figure 1.2: Systems connected in parallel

on a hybrid system. When a proxy server forwards an HTTP request upstream because all of the responses to the queries sent to the siblings are negative ones, it can choose one of the sibling proxies that returned the response fastest to the query as the next parent server. In this case, some scheme that avoids HTTP requests to be forwarded permanently is required.

Many researchers have worked hard to hatch out more efficient algorithms, more scalable architectures, and implementations with higher performance of distributed caching systems. Internet cache protocol (ICP) [3] is a protocol widely employed by the distributed WWW caching systems and currently it gains a position of de-facto standard. Using ICP, the proxies scattered all over the world can cooperate easily with each other and increase their cache hit rate. On the other hand, the traffic generated by ICP might be considered as a waste of network resources because it does not usually convey any WWW object data. Despite this disadvantage, ICP has been employed even in inappropriate situations because there is no quantitative analysis of the ICP traffic. Obviously, a system analysis based only on the hit rate is inadequate to evaluate the effect of the distributed WWW caching system.

Categorizing the distributed cache systems from a different viewpoint, ICP is a loosely coupled distributed caching system, i.e., caching nodes work independently of each other. On the other hand, tightly coupled systems where each caching node is assigned a portion of name space of WWW objects and caches only the objects whose names are in the assigned space. Hash routing [4] is a convincing algorithm for a tightly coupled caching system that achieves a high hit rate by preventing overlaps of

objects between caches. A drawback of hash routing, however, is its weakness against failure. When one of the caching nodes fails, all the users using the system suffer from the performance degradation.

1.2 Contributions of This Dissertation

This dissertation focuses on distributed WWW caching systems composed by scattered caching proxies on the Internet from various aspects such as network traffic, architectural scalability and fault tolerance. Especially the contributions of this dissertation include the following.

- Quantitative evaluation of the traffic on a distributed caching system
- Distinct guideline to decide the proper and improper situations for adoption of ICP
- Improvements of fault tolerance and efficiency of hash routing by introducing flexibility and redundancy

1.3 Contents of This Dissertation

This chapter, Chapter 1 gives a brief introduction of this work. It includes motivation, problems to be solved, and contributions.

Chapter 2 gives quantitative traffic analyses of distributed WWW caching systems especially focused on local area networks (LANs). From a viewpoint of LAN, ICP simply increases the ICP traffic passing through the border gateway that connects the LAN and the rest of the world. Because the network bandwidth between them is highly limited in many cases, ICP can cause a severe performance degradation. In our analysis, we figure out the communication model of the system and investigate the influence of several parameters on the traffic. As a conclusion of this analysis, we give a distinct configuration guideline of a distributed caching system using ICP.

In Chapter 3, the HTTP traffic on the WAN with a distributed caching system is focused on. From a viewpoint of WAN, ICP can actually decrease the HTTP traffic on the Internet by retrieving HTTP objects not from distant origin servers but from possibly closer proxy servers. However, the ICP traffic on the Internet might cancel the traffic reduction. In this analysis, we introduced the concept of network distance between caching proxies and origin servers. Thus, the effectiveness of ICP can be evaluated quantitatively.

Chapter 4 proposes a *duplicated hash routing* algorithm that achieves high tolerance against the failure of caching nodes. Duplicated hash routing introduces minimum

redundancy to keep system performance when some caching nodes are crashed. In addition, each node can cache objects requested by its local clients (*local caching*), which may waste some portion of system cache capacity but it can cut down the network traffic between caching nodes. We evaluate various aspects of the system performance such as hit rates, error rates and network traffic by simulations and compare them with those algorithms.

Finally, Chapter 5 concludes this dissertation and Chapter 6 suggests future works. As a supplement of our argument, we show how IP packets are exchanged between a client and server using HTTP, and between a proxy and a sibling using ICP in Appendix A. In Appendix B, we show the typical WWW traffic patterns using an actual log data generated by Squid cache system operated at our institute. All our analysis is based on these traffic models.

Chapter 2

ICP Traffic on LAN

A distributed WWW caching system based on Internet Cache Protocol (ICP) is a system where a cache of a single proxy server is shared by others. When a proxy server receives an HTTP request for an object but it does not store the object locally in its cache, it sends queries to other proxies. However, the system causes a problem that many ICP messages, i.e., many ICP packets, are exchanged between proxies. The increase of ICP packets cause various problems such as network congestion and high load on routers.

In this chapter, we focus on the influence of a distributed WWW caching system using ICP on network traffic of a LAN. we figure out the communication model of the system and analyze the traffic, especially the number of packets, on the LAN. With our analysis tracing the log data of an actual system, we reveal that ICP increases substantially the number of packets on a border gateway. Finally we conclude our argument with guidelines on organizing the distributed WWW caching system.

2.1 Introduction

One of the popular caching proxy servers used to form a globally distributed caching system is Squid [5]. When a Squid proxy server receives a request for an object it has not stored locally in its cache, it sends queries to other caching proxy servers. If one of the proxies has the requested object in its cache, the request is forwarded to the proxy and the object is retrieved from the proxy. The communication protocol to send queries to other proxy servers, called *siblings*, is Internet Cache Protocol (ICP) [3]. Because ICP uses UDP and is designed as simple, ICP is considered an efficient protocol.

Cache hits on a distributed caching system are classified into two kinds. One is a *local hit* and the other is a *remote hit*. An operator of a caching proxy server tends to set up the proxy to send ICP queries to many siblings for achieving a higher remote hit rate. In this case, many ICP messages are sent on small IP packets at a burst when

an HTTP request makes a local cache miss. However, the remote hit rate is much lower than the local hit rate. Since the caches of browsers and the proxy filter out the requests for frequently accessed objects, most ICP queries are ones for objects which are rarely requested. Therefore, most of these ICP messages can be considered as a waste of network resources.

A large number of ICP packets increase loads of routers, especially the border gateway between LAN and WAN where all the ICP packets are relayed. The load of the router depends on not only the total traffic volume but the total number of packets it processes. In addition, many organizations connect their LANs to the Internet with low bandwidth. Consequently, numerous ICP packets force a high load on a border gateway.

In past studies, many researchers argued about a hit rate [6, 7] and traffic volume [8, 9, 10]. We took a new approach: estimation of the number of packets which caching proxies and origin servers exchange with each other. In this paper, we introduce a communication model of a distributed caching system and evaluate the effect of several configuration factors. Furthermore, we investigate the number of packets based on the log data of actually working system. Finally we give some suggestions on configuring and managing a distributed caching system.

2.2 Why do We Estimate the Number of Packets?

Although ICP is considered to have few problems from a viewpoint of the amount of traffic because of its simple and small messages, it can make a high load on routers. A router has two time-consuming tasks to process a packet. One is a routing table look up for the next hop to deliver the packet. The other is copying the packet from the buffer of the incoming interface to that of the outgoing interface. Therefore, a large number of packets give a high load to the router even if the size of each packet is small. To make matters worse, the remote hit rate is usually low. For these reasons, it is important to examine the number of exchanged packets and make efforts to reduce them.

To obtain the number of the packets, there are two approaches: estimation and monitoring. We face, however, difficulties when we actually observe the number of packets. In order to show the effect of configuration parameters of caching proxies upon the number of packets they exchange, the observations must be done for many times changing the parameters. However, frequent configuration changes are not allowed at the systems that actually provide service to users. Even if such configuration changes are allowed, the HTTP requests the proxy receives and the state of its cache are changing every moment, therefore each observation is inevitably made in different condition. These are the reasons why we take an estimation approach.

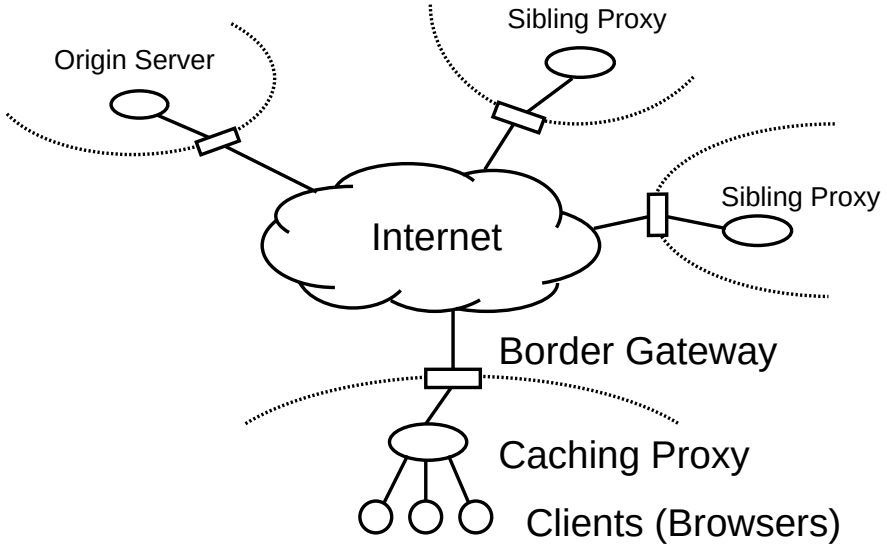

Figure 2.1: Network Environment

2.3 Modeling

As preparation for our estimation, we introduce a model of message flow on the system. Based on this flow model, we estimate the number of packets.

2.3.1 Network Environment

Figure 2.1 depicts the typical network environment. A caching proxy server is settled on the LAN of an organization. Many clients are connected to the LAN and they use the caching proxy server to access WWW servers on the Internet. This proxy exchanges ICP messages with several siblings on the Internet. In this study, we focus on the number of packets which pass through the border gateway since it relay all the ICP messages from or to the proxy on the LAN.

2.3.2 Flow of HTTP Requests and ICP Queries

In this subsection, we describe the flow model of HTTP and ICP messages. As Figure 2.2 shows, HTTP requests from a set of clients $C = \{C_1, ..., C_w\}$ are sent to the caching proxy P. If the proxy P has the requested object in its local cache, no more messages are generated outside the organization. Otherwise, the proxy P sends ICP queries to the set of siblings $S = \{S_1, ..., S_u\}$ and each of them returns reply, a hit or a miss, to the proxy P. If some siblings return hits, P forwards the HTTP request to

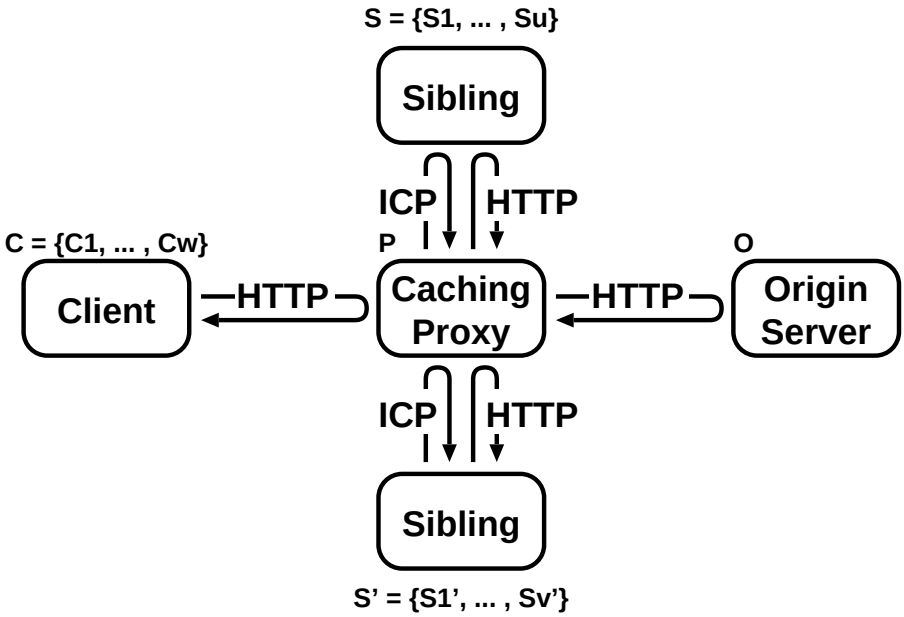

Figure 2.2: Flow of HTTP requests and ICP queries

the sibling that returns the hit first. If no siblings return hits, P forwards the HTTP request to its origin server O.

Independent of the flow of these HTTP and ICP messages, the proxy P receives ICP queries from a set of siblings $S' = \{S'_1, ..., S'_v\}$. We call these siblings *external siblings*. If the first hit response they receive is from the proxy P, they forward the HTTP request to P.

In our model, all the messages except for HTTP messages exchanged between the clients C and the proxy P are relayed on the border gateway.

2.3.3 Estimation of the Number of Packets

Based on our model, we formulate the number of packets which pass through the border gateway. Let N be the number of all packets which pass through the gateway during a certain time T. We classify these N packets into two categories. The first is for the packets generated to process HTTP requests from the clients on the organization network. The second is for the packets generated to process ICP queries and HTTP requests from the external siblings on the Internet. N_I and N_E denote the numbers of packets fall into the former and the latter category, respectively.

The Number of Packets N_I

Table 2.1: list of parameters (part 1)

R	the number of HTTP requests which the proxy P receives from clients during T
p	the local hit rate of the proxy P for HTTP requests
q	the remote hit rate of the proxy P for HTTP requests
u	the number of siblings to which P sends ICP queries
n	the average number of packets generated in a single HTTP transaction
m	the average number of packets generated in a single ICP transaction

Table 2.2: list of parameters (part 2)

R'_i	the number of HTTP requests sent to the external sibling S'_i during T $(i = 1, ..., v)$
v	the number of external siblings which send ICP queries the proxy P
p'_i	the local hit rate of the external sibling S'_i $(i = 1, ..., v)$
q'_i	the remote hit rate of the external sibling S'_i $(i = 1, ..., v)$
u'_i	the number of siblings to which the external sibling S'_i $(i = 1, ..., v)$ sends ICP queries
k_i	the rate of remote hits from the proxy P in all the remote hits of the external sibling S'_i $(i = 1, ..., v)$

First, we formulate N_I, the number of packets generated to process the local clients' HTTP requests. We use the parameters described in Table 2.1 to obtain N_I.

The set of clients C send HTTP requests to the proxy P. Since the local hit rate of the proxy P is p, it sends ICP queries to u siblings with the probability of $(1-p)$. One of these ICP queries makes a hit at a sibling with the probability of q, and in such case the HTTP request is forwarded to the sibling. The probability that all the ICP queries make misses is $(1-p-q)$, and in such case the HTTP request is forwarded to the origin server. Consequently, we obtain N_I as follows:

$$\begin{aligned} N_I &= R\{(1-p)um + qn + (1-p-q)n\} \\ &= R(1-p)(n+um) \end{aligned}$$

The Number of Packets N_E

Next, we formulate N_E. We assume the proxy P is not a parent proxy of any other proxy. We take this assumption for the simplicity of our model. A parent may receive HTTP requests from other proxies even if it returns misses to the ICP queries. This behavior makes our model more complicated.

Table 2.3: list of fixed parameters

R, R'_i	depends on the number of clients or characteristics of the clients
n	depends on characteristics of the HTTP requests from clients (see Appendix)
m	depends on the specification of ICP (see Appendix)
p'_i, q'_i, k_i	depends on the configuration of the cache S'_i

In addition to the parameters described in Table 2.1, we use the extra parameters described in Table 2.2. The external sibling S'_i receives R'_i HTTP requests and its local hit rate is p'_i. If S'_i does not have the requested object in its local cache, it send ICP queries to u'_i siblings and one of the ICP queries is for the proxy P.

The remote hit rate of S'_i is q'_i. When ICP queries from S'_i make hits at its siblings, S'_i forwards the HTTP request to the sibling which returns the hit first. The conditional probability that P returns the first hit is k_i.

Now, we can obtain N_E:

$$\begin{aligned} N_E &= m\sum_{i=1}^{v}(1-p'_i)R'_i + n\sum_{i=1}^{v}(1-p'_i)R'_i q'_i k_i \\ &= \sum_{i=1}^{v}(1-p'_i)R'_i(m+nq'_i k_i) \end{aligned}$$

2.4 Evaluation of the Number of Packets on the Model

We evaluate the number of total packets on our model and investigate the effect of the changes in parameters to the number of packets. The total number of packets N is as follows:

$$\begin{aligned} N &= N_I + N_E \\ &= R(1-p)(n+um) + \sum_{i=1}^{v}(1-p'_i)R'_i(m+nq'_i k_i) \end{aligned}$$

Since one of our objectives is to reveal the influence of the configuration of the proxy P on the number of packets, we fix the parameters which are independent of the configuration of the proxy P. Table 2.3 describes the fixed parameters. The parameters p, u, and v get influenced by configuration changes of the proxy P.

We set the values of R and R'_i at 1×10^5. This is because the system that is in operation at our institute receives this order of HTTP requests from the clients. For n

Table 2.4: Fixed value of parameters

R, R'_i	n	m	p'_i	q'_i	k_i
1×10^5	18.35	2	0.4	0.1	0.25

and m, we adopt the values which we describe in Appendix, that is, n is 18.35 and m is 2. For p'_i and q'_i, it is reported that a local hit rate is about 30-50% and a remote hit rate is about 10%[1] in general [11] and our study based on log data of Squid systems which are in operation at several sites confirms it. Therefore, we substitute 0.4 and 0.1 for p'_i and q'_i respectively. The value of k_i depends on many factors such as the distance between the P and S'_i, the local hit rate of S'_i, the number of siblings of S'_i, and so on. Since our purpose of this investigation is to estimate the effect of the system configuration to the number of packets, we set the value of k_i at 0.25. We summarize the values of fixed parameters in Table 2.4. Now, We can obtain the value of N as below.

$$N = 1 \times 10^5\{(1-p)(18.35+2u)+1.47v\}$$

Here, we investigate the total number of packets in two cases. One is the case that the proxy P sends ICP queries to siblings but does not accept ICP queries from external siblings. In this case, the value of the parameter v is 0. The other is the case that proxy P both sends and accepts ICP queries. In that case, the value of the parameter v is larger than 0. We argue both cases respectively.

2.4.1 Case 1: Proxy P does not Accept ICP Queries

In this case, the proxy P does not accept ICP queries from external siblings. The external siblings does not give any benefit to the clients in the organization which owns the proxy P. Particularly at an organization which has low-bandwidth connectivity to the Internet, the server administrator prefers to give high priority to local users on bandwidth utilization. In such an organization, it is agreeable to prohibit accesses from the external siblings.

Table 2.5 shows our estimation result. The total number of packets increases as the number of siblings to which P sends ICP queries increases. The increase rate gets high when the local hit rate is low. This indicates importance of server tuning. Supplementing the low local hit rate with remote hits by increasing the number of siblings makes the load of the border gateway high.

[1] This value is slightly higher than that of our observation.

Table 2.5: The value of N: case 1 (shown in millions)

$p \setminus u$	0	2	4	6	8	10
0.3	1.28	1.56	1.84	2.12	2.40	2.68
0.4	1.10	1.34	1.58	1.82	2.06	2.30
0.5	0.92	1.12	1.32	1.52	1.72	1.92

Table 2.6: The value of N: case 2 (shown in millions)

$p \setminus u$	0	2	4	6	8	10
0.3	1.28	1.86	2.43	3.01	3.58	4.15
0.4	1.10	1.64	2.17	2.70	3.24	3.77
0.5	0.92	1.41	1.91	2.40	2.89	3.39

2.4.2 Case 2: Cache P Accepts ICP Queries from External Siblings

In this case, the proxy P both sends and accepts ICP queries. For example of this case, an organization connected to the Internet with high network bandwidth can provide WWW caching service to external siblings. A distributed WWW caching system where several proxies send ICP queries to each other is also the case.

Here, we take one more assumption that the number of the siblings v and the number of the external sibling u are same. Under this assumption, several proxies make a group and they all sends ICP queries to each other.

The estimation result is shown in Table 2.6. The increase rate of the total number of packets is much higher than that in the case of $v = 0$. Therefore, a server operator in an organization where the load of the border gateway is an important issue has to be cautious about accepting ICP queries form external siblings.

2.5 Evaluation of the Actual Proxies

In this section, we evaluate the number of packets based on our model and access log data of a Squid proxy server system which was in operation at our institute for a month in May 1998. The average values of parameters in a day derived from the access log are shown in Table 2.7. Since the number of ICP queries this proxy accepted is very small, we can neglect N_E. Therefore, we can consider this proxy is an example of the case 1 described in subsection 2.4.1.

Since over 10% improvement on local hit rate through server configuration tuning

Table 2.7: The values of parameters

R	p	u	q	n	m	N_I	N_E	N
106545	0.34	8	0.065	18.35	2	2415481	3784	4312464

Table 2.8: The number of packets (shown in millions)

$p \setminus u$	0	2	4	6	8
0.34	1.29	1.57	1.85	2.13	2.42
0.39	1.19	1.45	1.71	1.97	2.23
0.44	1.09	1.33	1.57	1.81	2.05

has little reality, we evaluate the total numbers of packets respectively in the cases that the increase of local hit rate is 0%, 5%, and 10%. We also change the number of siblings and evaluate the number of packets in each case. The result is shown in Table 2.8.

From this result, we can point out that our proxy achieves remote hit rate of 6.5% although the number of packets increases from 1.29 millions to 2.42 millions (87% increase). One of the approaches to reduce the total number of packets is to increase the local hit rate. A 10% increase in the local hit rate cuts off 16% of total packets. This is one instance that shows importance of system tuning.

Next, we consider the effect of siblings. This proxy sends ICP queries to 8 siblings and the remote hit rates $q_i (i = 1, \cdots, 8)$ are 0.031, 0.013, 0.010, 0.006, 0.002, 0.001, 0.000, 0.000, respectively. Figure 2.3 shows the relation between the total hit rate $(p + q)$ and the total number of packets N. In this graph, the number of siblings u works as a parameter, and we adopt siblings in order of higher remote hit rate as u increases. We can conclude that many ICP messages waste network resources and give a high load to the border gateway. Clearly, it is a good idea to cease sending ICP queries to the siblings that rarely return hits.

2.6 Concluding Remarks

Distributed caching systems which use the ICP protocol come into wide use. ICP is considered to be simple and efficient. However, there is a problem that numerous packets caused by ICP increases the load of border gateways.

To clarify this problem, we estimated the total number of packets sent to the network and showed that the more siblings a caching proxy sent ICP queries to, the more packets were exchanged on the network. The increase rate of the number of packets is high when the local hit rate is low. This situation gets worse when the

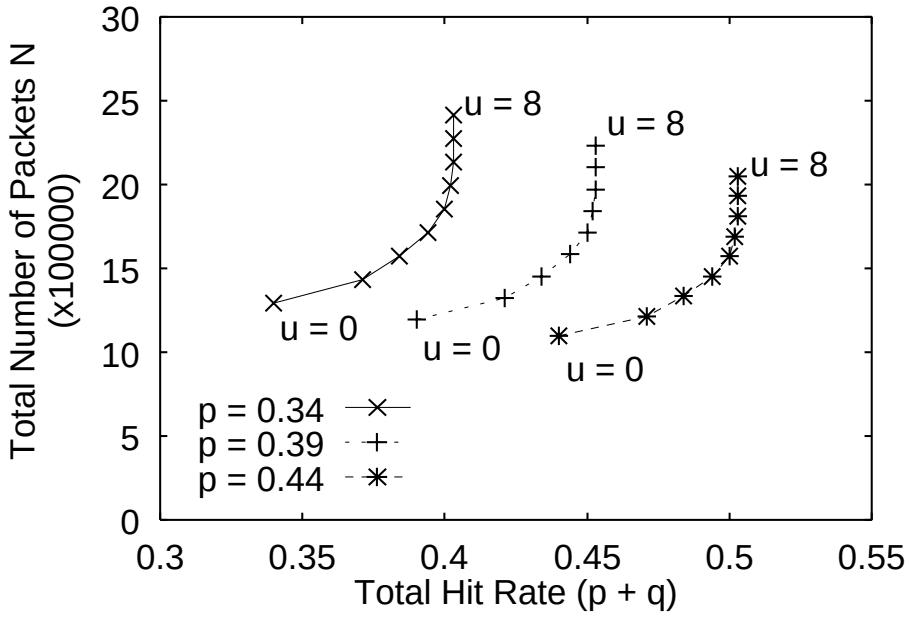

Figure 2.3: The relation between hit rate and the number of packets

proxy accepts ICP queries from external siblings. Our analysis based on the actual log data made it clear that many ICP packets wasted network resources.

Finally, we conclude our argument with suggestions on configuring a distributed caching system, especially on making the load of border gateways low.

Minimize the number of siblings, especially, external siblings: Since HTTP requests are forwarded to the sibling which returns the hit to the ICP queries first, distant siblings may not contribute to the remote hit rate even if they return a lot of hits to ICP queries. Consequently, we should check the remote hit rate of each sibling. Moreover, ICP queries from external siblings never contribute to the clients of the organization and drastically increase the number of packets relayed at the border gateway. We should count the number of relayed HTTP requests from each external sibling, which indicates the contribution to the remote hit rate of the external sibling.

Improvement of the local hit rate is the most important: Increase of the local hit rate decrease the number of HTTP requests and ICP queries. Configuring the cache parameters (cache size in memory or disks, time out limit, etc) to match the characteristics of HTTP requests is important.

Chapter 3

ICP Traffic on WAN

In this chapter, we discuss the scalability issues arising in distributed WWW caching systems that exchange meta information among caching proxies using Internet Cache Protocol (ICP). One of the main objectives of WWW caching systems is to reduce WWW traffic on the Internet. Since the traffic generated by ICP does not usually convey WWW objects, it might be considered as a waste of network resources. However, ICP is expected to improve the hit rate so that the system would in turn reduce WWW traffic on the Internet. Obviously, a system analysis based only on the hit rate is inadequate to evaluate the effect of the distributed WWW caching system. Therefore, it is indispensable to examine quantitatively how much the traffic is reduced or conversely increased by the system. For the quantitative analysis of the system effect, we evaluate the amount of overall traffic generated by ICP and HTTP in the system with siblings as well as in the system without siblings. From the results, we derive the requirements of the system for substantial traffic reduction. Considering the fact that the remote hit rate of a single sibling proxy is usually very low, we conclude that the distributed WWW caching system using ICP can hardly contribute to the reduction of WWW traffic.

3.1 Introduction

One of the main goals of WWW caching system is to reduce WWW traffic on the Internet. Since a local cache hit at a caching proxy does not produce any further traffic between the proxy and the origin server, the local cache hit has an advantage in the amount of traffic generated on the Internet. On the other hand, if the proxy does not have the requested object in its local cache, ICP messages, i.e., ICP queries and ICP replies, are exchanged over the Internet. The ICP messages usually do not convey any WWW objects so that they are just overhead of the system from the viewpoint on the amount of traffic. As mentioned in Chapter 2, the remote hit rate produced by

ICP messages is very low in many cases. The number of accesses to WWW pages as a function of its access ranking is well known to follow the Zipf distribution [12, 13], which indicates that most objects are accessed only once and are never accessed after that. In other words, most objects cannot be shared among the proxies. Consequently, most of the ICP messages can be considered as a waste of network resources from the viewpoint of the Internet.

According to these complex circumstances, we can conclude that the hit rate is not adequate for the evaluation of distributed WWW caching systems. In fact, even if the total hit rate increases due to remote hits, it is likely that ICP generates more traffic than the traffic reduced by remote hits.

In this chapter, we focus on the total amount of traffic over the Internet generated by a distributed WWW caching system. In general, communication cost is evaluated by means of traffic volume or number of packets. For the evaluation of total traffic on the Internet, however, we define new indices which allow for hop count. By considering hop count, we can evaluate total communication cost quantitatively. The goal of this paper is to make it clear that the distributed WWW caching systems using ICP can hardly decrease the network resource utilization on the Internet.

In Section 3.3, we introduce Packet-Hop Index (PHI) and Byte-Hop Index (BHI) as measures of communication cost. In Section 3.4, we describe a model of the communication environment. The characteristics of WWW traffic on the Internet are summarized in Section 3.5. Finally, we evaluate the distributed WWW caching system and discuss several design issues arising there.

3.2 Related Works

Several works closely related to the performance evaluation of the WWW system including the caching proxy services have been done so far. However, there are few reports covering the proxy caching systems.

Some researchers have argument that ICP degrades the system performance [14, 15]. However, they only pointed out that ICP added an extra RTT to the object retrieval time in a case of a remote cache miss. Since WWW users usually mind how much service delay of WWW accesses is, its reduction will benefit them. However, the extra RTT is not so significant for WWW users because the extra RTT is only in the order of hundreds milliseconds. Normally WWW users are irritated by the service delay which is in the order of seconds or more. Therefore, the performance evaluation mentioned in [14, 15] is not enough to reveal reasons why the service delay is quite large. Furthermore, benefits and/or drawbacks of the caching proxy servers are still not clear.

In general, the network performance is going to degrade drastically in the situation where the network is highly loaded, as mentioned in [16]. Today, the Internet can

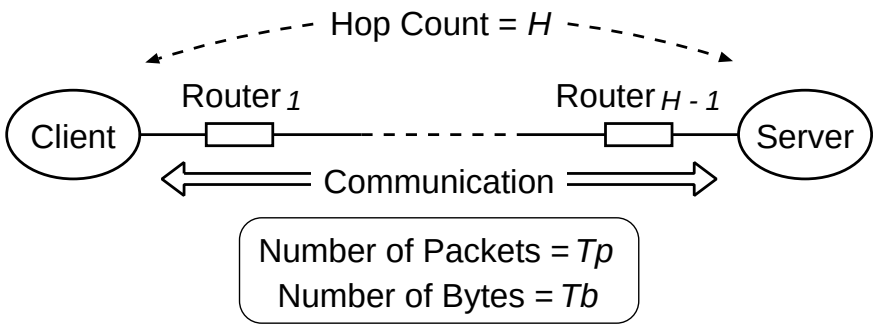

Figure 3.1: Basic model of a communication

be considered as a highly loaded network because of the rapid growth of the WWW services on the Internet. Therefore, we can suppose that reducing the WWW traffic can help much on the decrease of the service delay. Originally, the caching proxy system as well as the ICP is designed as a simple, small, and efficient system for reducing the WWW traffic over the Internet [17]. However, there are few reports on the analysis of how much the caching proxy system can reduce the WWW traffic precisely. These are the reasons why our research is focused on both the quantitative analysis of ICP traffic and the total traffic reduction by the caching proxy systems.

3.3 The Definition of PHI and BHI

In this section, we introduce two indices, Packet-Hop Index (PHI) and Byte-Hop Index (BHI). Figure 3.1 gives a basic model of a *communication* between two entities: a client and a server. In this model, H denotes the hop count between the client and the server. The hop count is defined as the number of network segments in the route from the client to the server. In a single communication, we assume T_p packets are exchanged between the client and the server, and the total amount of exchanged data is T_b bytes. With this model, we can define PHI and BHI of the communication between the client and the server as follows:

$$PHI = T_p H$$

$$BHI = T_b H$$

Next, we define a *task*. For example, a single access to a WWW object consists of ICP message exchanges and several HTTP sessions. With this observation, we can model that a *task* (WWW object access) is a set of *communications* (ICP message exchanges and HTTP sessions). As the definition of PHI and BHI for a communication,

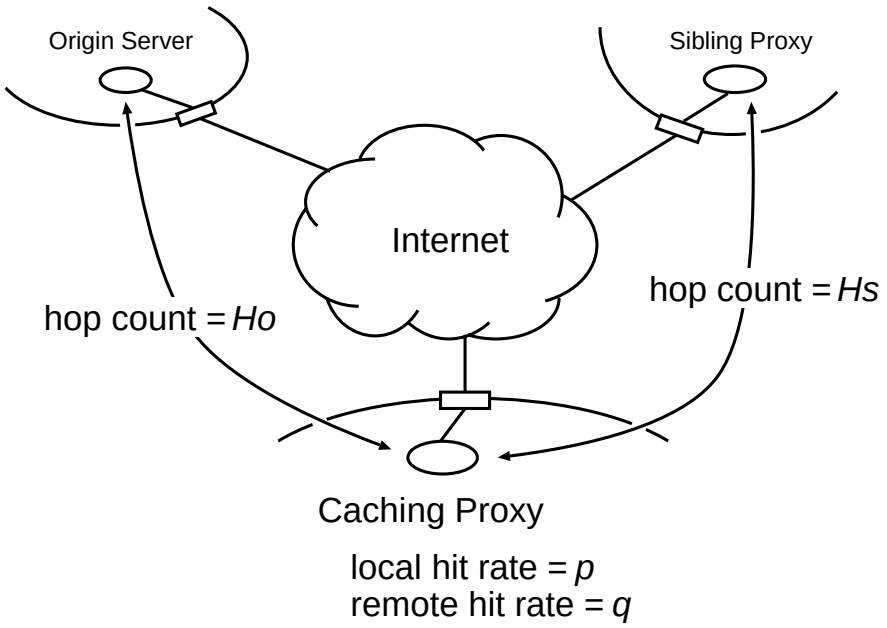

Figure 3.2: Communication environment of a distributed WWW caching system

we expand the definition to both PHI and BHI of a task; the PHI and BHI of a task are defined as the sum of PHI and BHI of each communication, respectively.

$$PHI_T = \sum_i PHI_i$$

$$BHI_T = \sum_i BHI_i$$

3.4 Communication Models

In this section, we describe the communication model of a distributed WWW caching system. Our simple model depicted in Figure 3.2 consists of three components: a caching proxy server, a sibling proxy server, and an origin server. When the caching proxy receives an HTTP request for the object which is not kept in the local cache, the caching proxy sends an ICP query to the sibling proxy. If the sibling proxy has the requested object in its cache, the sibling proxy makes an ICP reply for indicating that the sibling proxy has the object. In this case, the caching proxy forwarded the HTTP request originally sent by the client to the sibling proxy to obtain the object. When any sibling proxies do not have the requested object locally, however, the HTTP request is forwarded to the origin server.

Here, we let H_o be the hop count from the caching proxy to the origin server. Also, H_s denotes the hop count from the caching proxy to the sibling proxy. We suppose p and q are the local hit rate and the remote hit rate of the caching proxy, respectively. More precisely, q is the ratio of the HTTP requests for the objects which make cache hits at the (remote) sibling proxy to all the HTTP requests received by the caching proxy.

With these definitions mentioned above, we define a *task* of a caching proxy as a series of *communications* necessary to retrieve a requested object. There are three cases on the WWW object retrievals: (1) a local hit in the caching proxy, (2) a remote hit at the sibling proxy, and (3) a remote miss at the sibling proxy. In the case (1), the caching proxy returns the requested object locally, therefore, no further communications are required. In the case (2), two communications between the caching proxy and the sibling proxy are required: one is for an exchange of ICP messages (a query and a reply), and the other is to obtain the object from the sibling proxy through the HTTP connection between the caching proxy and the sibling proxy. In the last case (3), there are two communications required: one is for ICP with the sibling proxy, and the other is for HTTP with the origin server. Note that we do not discuss the case where multiple sibling proxy servers are configured for the caching proxy server. It is popular to use the multiple sibling proxy servers, however, it can be considered as just an additional overhead on the process of ICP message handling. In this case, communications corresponding to each sibling proxy are simply added cumulatively. Hence, the PHI and BHI may be larger than ones in the case where only a single sibling proxy is used.

3.5 Summary of Characteristics of WWW Traffic

As a basis for the quantitative analysis of the caching proxy system, we examined the statistical characteristics of the WWW traffic. The statistical analysis shown in this section is based on the log data generated by Squid caching proxy system which was in operation at our institute for a month in May 1998. During this period, the local hit rate of our Squid was 34% and the total remote hit rate was 6% with the configuration where the number of siblings was 8. The details of this analysis are also shown in Appendix.

3.5.1 Object Size

For estimations of Tp and Tb, we carefully did examinations on the size of WWW objects and the size of both ICP messages and HTTP requests. We observed that the average size of WWW objects handled by our Squid was 7356 bytes and its median was 1460 bytes. For our further analysis, we classified the WWW objects into four

Table 3.1: Hit rate, average object size and average number of packets

result	rate	object size	number of packets
local hit	34.3 %	4606 bytes	14.84
local miss	65.7 %	8790 bytes	18.21
remote hit	6.5 %	6958 bytes	16.92
all miss	59.2 %	8993 bytes	18.35

categories based on their cache status, and examined the average size for each category. In case of *local hit*, the average size of the objects obtained from the local proxy was 4606 bytes. In case of *local miss*, which means the local proxy does not have the requested object in the local cache, the average was 8393 bytes. In case of *remote hit*, which means the local proxy does not have the requested object but the sibling proxy has it, the average was 6958 bytes. In case of *all miss*, which means the local proxy as well as the sibling proxy does not have the requested object, the average size of the objects in this category was 8538 bytes. In this paper, B_{LH}, B_{LM}, B_{RH}, and B_{AM} denote these average size, respectively.

The other important number we have to know is the average size of ICP messages. Since there is no information on the size of ICP messages in Squid's log data[1], we had to estimate the size based on the retrieved URL. In brief, a single ICP message defined in [3] has its header and the payload in which a retrieved URL is stored as an ASCII text. With this observation, we can derive that the average size of the ICP messages is about 70 bytes.

3.5.2 Number of Packets

The data exchanged between a client and a server is divided into several packets. The number of packets, however, depends on many factors: (1) implementation of protocols such as ICP, HTTP, TCP, UDP, and IP, and (2) underlying datalink technologies, and (3) the conditions of the network such as congested or not. We've make a rough

[1] Squid can record only the size of ICP reply messages corresponding to the ICP queries that the Squid receives.

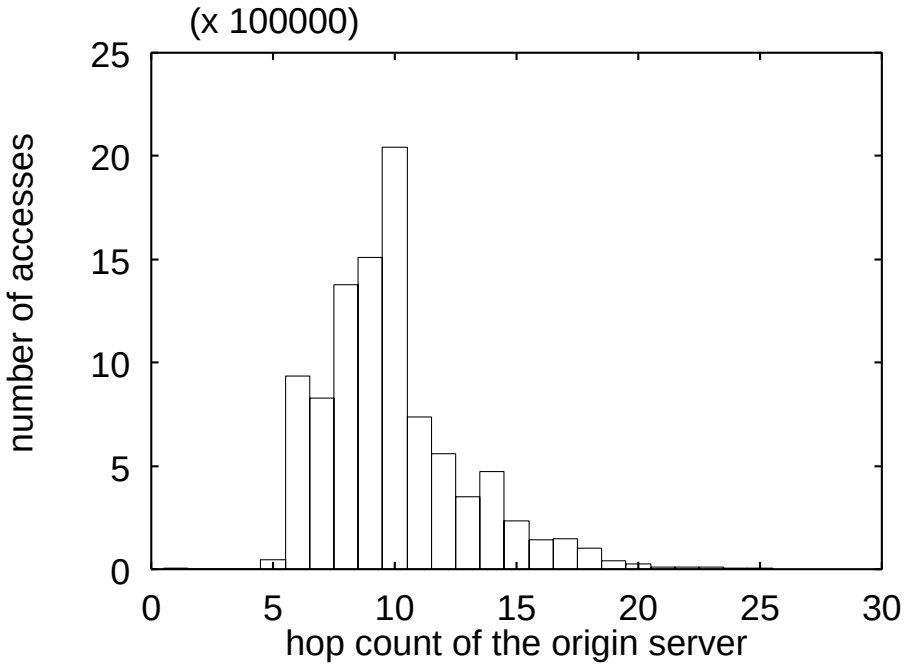

Figure 3.3: Distribution of hop counts from our proxy to origin servers

estimate of the number of packets P for the object size B as follows:

$$P = \begin{cases} 11 & (0 < B \leq 1460) \\ 13 & (1460 < B \leq 2920) \\ 14 & (2920 < B \leq 4380) \\ 16 & (4380 < B \leq 5840) \\ 17 & (5840 < B \leq 7300) \\ 18 & (7300 < B \leq 8760) \\ 18 + \left\lceil \left\lceil \frac{B-8760}{1460} \right\rceil * \frac{5}{4} \right\rceil & (8760 < B) \end{cases} \tag{3.1}$$

Using this estimation, we can calculate the average number of packets exchanged for a retrieval of a single WWW object over HTTP. In case of local hit, local miss, remote hit, and all miss, the average number of packets is 14.87, 18.21, 16.92 and 18.35 respectively. In this paper, P_{LH}, P_{LM}, P_{RH} and P_{AM} denote these average numbers of packets, respectively. The summaries of the results are shown in Table 3.1.

3.5.3 Hop Count

Figure 3.3 shows the distribution of the hop counts from our institute to origin servers measured by `traceroute` program. In this measurement, we listed up origin servers

from Squid's log data, and tried to apply `traceroute` for each origin server. Because several WWW servers do not permit ICMP probes by some security reasons, our measurements for more than 20% of origin servers failed. Figure 3.3 does not include any measurements for this kind of WWW servers.

Obviously, there is an argument that the hop counts to each origin server may vary, because there are several alternative routes from our institute to the origin server. However, the hop counts we measured can be considered accurate enough to make our analysis, because of the stable operation of the Internet. It is not likely to change the count frequently. Note that the distribution of hop counts to origin servers from any other sites may change drastically. In spite of a certain level of inaccuracy, we can find a tendency in the distribution of hop counts; the most hop counts range roughly from 5 to 20, and its mode is 10.

3.5.4 Evaluation of PHI and BHI on Systems

The goals of our performance evaluation of the distributed WWW caching system are to examine whether the WWW caching proxy system actually reduces the WWW traffic on the Internet and to estimate the threshold where benefit produced by the system exceeds the communication overhead for WWW cache system itself. For these goals, we apply both PHI and BHI to tasks we observed.

With our observation on our Squid, we can assume that there is no correlation between the hop counts to the origin servers H_o and the distribution of the packet size P as well as the object size B. In other words, this assumption means that the distributions of both P and B do not depend on H_o. With this assumption, we are going to use the average PHI_T and BHI_T of tasks whose objects are retrieved from origin servers with the hop count H_o as the performance index of the WWW caching proxy system.

$$\overline{PHI}(H_o) = \underset{H_o}{\text{Average}}(PHI_T)$$

$$\overline{BHI}(H_o) = \underset{H_o}{\text{Average}}(BHI_T)$$

3.6 Analysis

In this section, we discuss our quantitative analysis on the cost associated with communications among a caching proxy, a sibling proxy, and an origin server in the distributed WWW cache architecture. Our approach is to use both PHI and BHI. Furthermore, we derive the boundary conditions where the benefit produced by the system exceeds the communication overhead for WWW cache system. Based on these analyses, we point out the drawbacks of the distributed WWW cache system using ICP.

3.6.1 Analysis Using PHI

As this article is focused on the benefit/drawbacks of the distributed WWW caching proxy system, we apply the PHI to two cases: a caching proxy with a sibling proxy and one without a sibling proxy.

Without a Sibling Proxy

In the case with no sibling proxy, all the HTTP requests that make local cache misses are forwarded to their origin servers. Therefore, there is no overhead for handling ICP messages. We define P_{LM} as the average number of packets exchanged between the caching proxy and the origin server. The PHI can be formulated as follows:

$$\overline{PHI}(H_o) = (1-p)P_{LM}H_o \tag{3.2}$$

With a Sibling Proxy

A caching proxy with a sibling proxy acts more complicated. When an HTTP request misses the local cache at the caching proxy, an ICP query is sent to a sibling proxy. Since the average size of ICP queries is about 70 bytes as mentioned in Section 3.3, a single UDP packet can carry each ICP query. The ICP query makes the remote hits at the sibling proxy with the probability q. In this case, the HTTP request is forwarded to the sibling proxy. If the ICP query makes a miss, on the other hand, the HTTP request is forwarded to the origin server. The average number of packets exchanged between the caching proxy and the origin server is defined as P_{AM}. Therefore, the PHI is derived as follows.

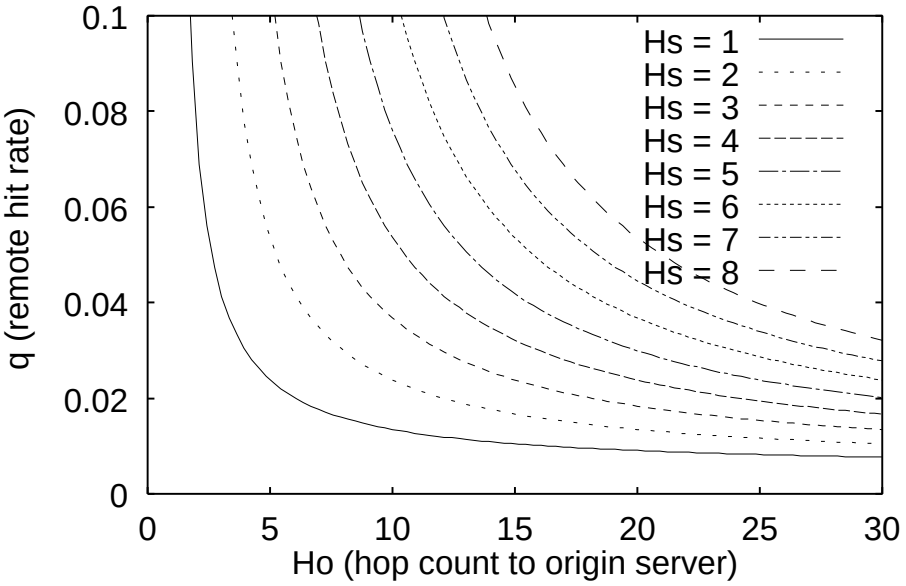

Figure 3.4: Threshold (PHI)

$$\begin{aligned}\overline{PHI}(H_o) &= (1-p)(2\times 1)H_s + qP_{RH}H_s \\ &\quad + (1-p-q)P_{AM}H_o \\ &= (2(1-p)+qP_{RH})H_s \\ &\quad + (1-p-q)P_{AM}H_o \end{aligned} \tag{3.3}$$

Threshold

It is obvious that the communication cost for ICP messages is inversely proportional to the hop counts to the origin server. The underlying idea of the sibling proxy is to obtain WWW objects from a proxy closer than the origin server in order to reduce the data transfer overhead. If the origin server is closer than the sibling proxy, there is no advantage to use the sibling proxy; getting WWW objects directly from the origin server is more efficient in terms of communication overhead. The remote hit rate is also an important factor. The higher the remote hit rate is, the more benefit the caching proxy can get. Therefore, from Equation 3.2 and Equation 3.3, we can derive the boundary condition where the ICP overhead is equal to the benefit of the sibling proxy:

$$\begin{aligned}(1-p)P_{LM}H_o &= (2(1-p)+qP_{RH})H_s \\ &\quad + (1-p-q)P_{AM}H_o\end{aligned}$$

We can get the minimum requirement of remote hit rate q by solving this equation for q.

$$q = \frac{(1-p)(2H_s + (P_{AM} - P_{LM})H_o)}{P_{AM}H_o - P_{RH}H_s}$$

As shown in Table 3.1, the value of p, P_{LM}, P_{RH} and P_{AM} to 0.34, 18.21, 16.92 and 18.35, respectively. With these numbers, we examine the relation between H_o and q. The H_s is varied from 1 to 8, as a parameter. Figure 3.4 depicts the relation.

$$q = \frac{0.66(2H_s + 0.14H_o)}{18.35H_o - 16.92H_s}$$

Note that the total remote hit rate more than 10% cannot be achieved even if we use the multiple sibling proxy configuration[2] in the ordinary environment. With our evaluation of log data of Squid operated at many sites, few sibling proxies achieve high remote hit rate more than 2% individually.

Figure 3.4 shows the break-even condition between q and H_o, in terms of communication cost based on the PHI. For example, considering WWW servers within 15 hops from the caching proxy, we can conclude that the sibling proxy whose sole remote hit rate is less than 2% has no advantage unless the sibling proxy is within 2 hops away from the caching proxy.

Next, we examine the impact of the local hit rate p to the performance; we vary the local hit rate p. The results are shown in Figure 3.5. Here we set H_s to 3 (the *minimum* hop count in the model shown in Figure 3.2). In general, the higher a caching proxy achieves the local hit rate, the lower the remote hit rate becomes. It is the most likely that we can't find out the WWW object in the sibling proxy if we can't find it in the local caching proxy, since the distribution of accesses to WWW objects follows the Zipf distribution as mentioned in Section 3.1. Therefore, even when we improve the local hit rate, it becomes more hard to fulfill the requirement on the remote hit rate.

As our conclusion, it is hard to achieve the traffic reduction on WWW accesses in the architecture of the WWW caching proxy with sibling proxies. Especially, it is not practical to use siblings in other organizations where their networks are normally far with the hop count more than 3.

3.6.2 Analysis Using BHI

Most of these evaluations can be done in the same way as those of PHI.

[2]In the case we observe the total remote hit rate more than 10%, we suppose that the (local) caching proxy is not configured appropriately or it is in an exceptional environment.

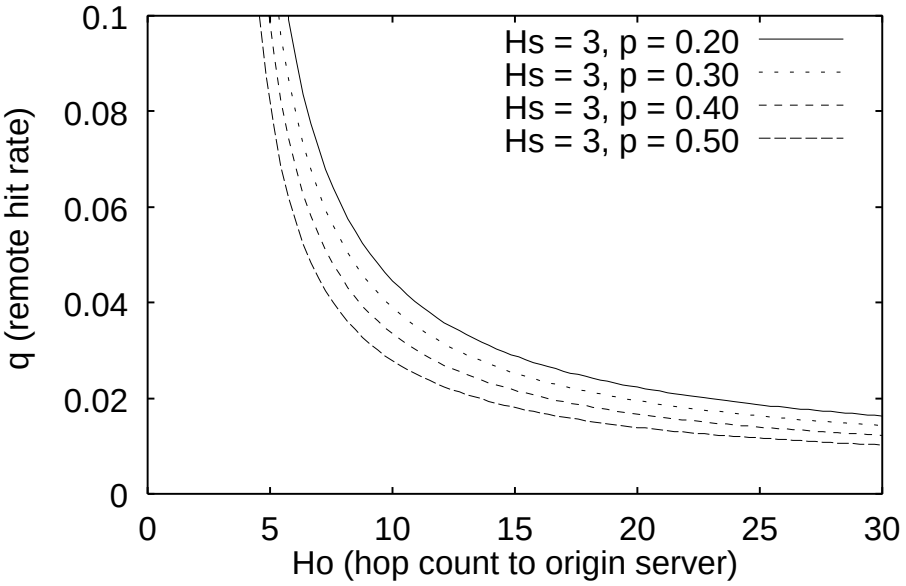

Figure 3.5: Threshold on $H_s = 3$ (PHI)

Without a Sibling Proxy

The average number of bytes exchanged between the caching proxy and the origin server in case of *local miss* is denoted as B_{LM}. The BHI is formulated shown below:

$$\overline{BHI}(H_o) = (1-p)B_{LM}H_o \tag{3.4}$$

With a Sibling Proxy

The average sizes of both ICP queries and replies are about 70 bytes, as mentioned in Section 3.5. To make the calculation simple, we fix them to 70 bytes. B_{RH} denotes the average number of bytes transferred between the caching proxy and the sibling proxy in case of *remote hit*. The BHI is formulated as below:

$$\begin{aligned} \overline{BHI}(H_o) &= (1-p)(2\times 70)H_s + qB_{RH}H_s \\ &\quad + (1-p-q)B_{AM}H_o \\ &= (140(1-p) + qB_{RH})H_s \\ &\quad + (1-p-q)B_{AM}H_o \end{aligned} \tag{3.5}$$

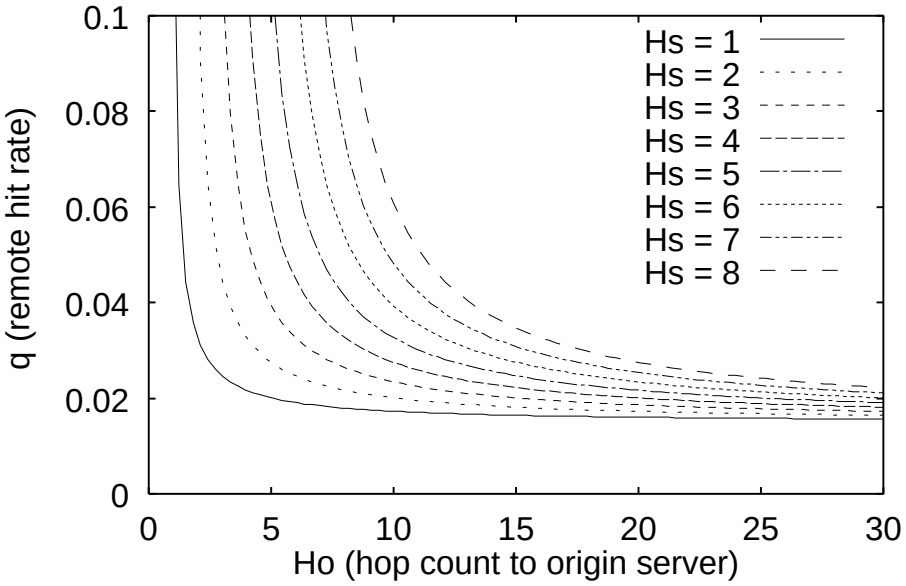

Figure 3.6: Threshold (BHI)

Threshold

Similar to the analysis on PHI, we derive the boundary condition. From Equation 3.4 and Equation 3.5, the boundary condition is expressed as follows:

$$\begin{aligned}(1-p)B_{LM}H_o &= (140(1-p)+qB_{RH})H_s \\ &\quad + (1-p-q)B_{AM}H_o\end{aligned}$$

Now, we can obtain the minimum requirement of the remote hit rate q by solving this equation for q.

$$q = \frac{(1-p)(140H_s + (B_{AM}-B_{LM})H_o)}{B_{AM}H_o - B_{RH}H_s}$$

As shown in Table 3.1, we can set the value of p, B_{LM}, B_{RH} and B_{AM} at 0.34, 8790, 6958 and 8993, respectively. H_s is a parameter varied from 1 to 8. Figure 3.6 shows the boundary conditions.

$$q = \frac{0.66(140H_s + 203H_o)}{8993H_o - 6958H_s}$$

In comparison with the case of PHI, the required remote hit rate q decreases rapidly as H_o increases. We can point out from this result that a sibling proxy whose remote hit rate is less than 2% has almost no advantage on reducing WWW traffic even if

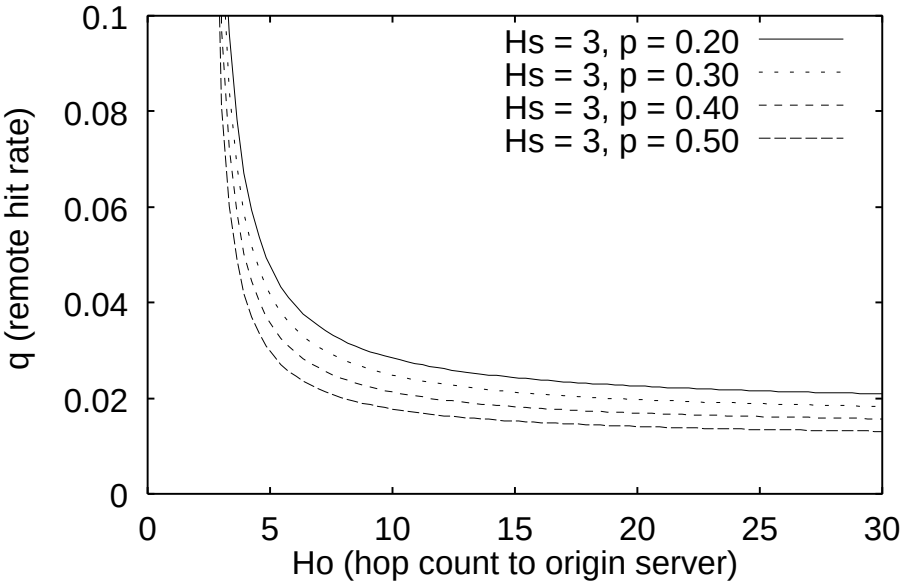

Figure 3.7: Threshold on $H_s = 3$ (BHI)

the sibling proxy is hooked up to the same network segment with the caching proxy. As mentioned in the previous subsection, sibling proxies rarely achieve the remote hit rate higher than 2%. Therefore, this result indicates that most of the sibling proxies cannot provide any benefit for a caching proxy, in terms of reduction of the WWW traffic.

Similar to the PHI analysis, we vary the local hit rate p, and figure out how much the boundary condition between H_o and q is changed, in Figure 3.7. In this analysis, the H_s is fixed at 3. From this analysis, even in the case that the local hit rate is 50%, the sibling proxy has to achieve the remote hit rate more than 2%. With the same reasons described in the case of PHI, we can conclude again that most of these siblings are useless and ICP messages can be considered completely as waste of network resources.

3.7 Concluding Remarks

The essential drawback of ICP is that ICP queries are invoked every time when any HTTP requests make *local miss* at the caching proxy. Though ICP uses UDP as its transport layer service and its message size is fairly small, it is too expensive to provide any improvements in terms of the WWW traffic reductions. RFC2187 [17] asserts the advantages of ICP such as:

- ICP can make reductions of the delay on the deliveries of WWW objects.
- ICP can be used as an indicator of the reachability to neighboring proxies.
- ICP can achieve the load balancing among sibling proxies.

Some of these would be correct, however, our analysis confirms that the performance advantages described in [17], especially on both traffic and delay reductions, cannot be achieved by almost all of the sibling proxies in the ordinary environment of the Internet.

Other approaches can be taken for making improvements on the distributed WWW caching proxy system. One of them is to use the fixed configuration for forwarding the HTTP requests, such as a hierarchical "tree" configuration among caching proxies. The purpose of ICP is to identify the hosts to which the HTTP requests are forwarded in case of local miss at the local caching proxy. Since the "on-demand" cache query scheme implemented as ICP can be considered as a waste of network resources, using the fixed configuration for forwarding the HTTP request might be a possible solution because of eliminating any uses of ICP. This approach can help on reduction of the ICP traffic, however, it is still too hard to achieve the remote hit rate higher than one using ICP. Consequently, it is more likely that the forwarded HTTP requests cause the cache misses at the remote proxy. Therefore, this approach cannot make much difference from the case of ICP.

The other approach is to use a "transparent" caching system. Recently, several networking equipment vendors released their new WWW cache products in which they are using the "transparent" caching scheme. Their basic idea is to sneak and cache the WWW transactions at the router. Such kind of system resides with a router where the system always monitors any HTTP requests and replies crossing through the router. If an HTTP reply including a WWW object, the system copies and puts the object into its cache. If the system finds out an HTTP request for an object stored in its cache, then the system intercepts the request and returns the object back to the client who claims the object. This approach can achieve the traffic localization and may reduce the overall traffic for WWW in the Internet, however, other kinds of issues may arise such as security issues, a waste of cache storage, etc.

The demands for the capacity of the backbone network have been increasing, and it is no exaggeration to say that this demand is caused by the growth of the WWW utilization. However, the actual backbone capacity is not sufficient for all the demand in the Internet. Accordingly, it becomes more significant that any schemes to make reduction on the WWW traffic are developed. As discussed in this article, caching proxy systems using ICP cannot contribute to the reductions of the WWW traffic. Therefore, it is necessary to develop a practical, scalable, and efficient cache mechanism for this goal.

Chapter 4

Duplicated Hash Routing

Hash routing is an algorithm for a distributed WWW caching system that achieves a high hit rate by preventing overlaps of objects between caches. However, one of the drawbacks of hash routing is its lack of robustness against failure. Because WWW becomes a vital service on the Internet, the capabilities of fault tolerance of systems that provide the WWW service come to be important. In this paper, we propose a *duplicated hash routing* algorithm, an extension of hash routing. Our algorithm introduces minimum redundancy to keep system performance when some caching nodes are crashed. In addition, we optionally allow each node to cache objects requested by its local clients (*local caching*), which may waste cache capacity of the system but it can cut down the network traffic between caching nodes. We evaluate various aspects of the system performance such as hit rates, error rates and network traffic by simulations and compare them with those of other algorithms. The results show that our algorithm achieves both high fault tolerance and high performance with low system overhead.

4.1 Introduction

The distributed WWW caching systems are put into two categories from a viewpoint of a role of each caching node: *loosely* coupled systems and *tightly* coupled systems.

In a loosely coupled system, each caching node caches all the objects requested by its *local* clients. A local client of a caching node means a client that sends requests directly to the node. If the caching node which receives a request does not have the requested object in its local cache, it tries to retrieve the object from the other nodes by some method. Internet cache protocol (ICP) [3] is one of the protocols used in this kind of systems.

In a tightly coupled system, each caching node is assigned a portion of name space of WWW objects and caches only the objects whose names are in the assigned space. All the requests for a WWW object are forwarded to a single caching node and cached

there. Because these systems optimize their whole storage capacity, they can achieve higher hit rates than loosely coupled systems. There are several systems put into this category and some of them are implemented [18, 19, 20, 4].

There are two major disadvantages in a tightly coupled system. One is its poor scalability. A tightly coupled system cannot be employed in an environment with low bandwidth between caching nodes. Advances in backbone network technology will make it possible to operate the tightly coupled caching system over a distributed area in the near future. However, even if such a desirable situation comes true, a kind of localization of communication, which reduces network traffic between caching nodes, is desirable.

The other disadvantage is its lack of robustness. Because each caching node has to handle the requests from all clients in the system, failure of a single node inevitably has influence on all the clients. This drawback can be removed by introducing redundancy to the system.

In this paper, we propose a *duplicated hash routing* algorithm, an extension to the simple hash routing [4]. Hash routing is a simple and efficient algorithm for tightly coupled caching systems. By introducing cache redundancy to the system, our algorithm can keep its high hit rate even when some caching nodes are in failure. The redundancy of our system is moderate, i.e., decrease of the hit rate and increase of the network traffic between caching nodes is small. In addition, we optionally enable *local caching* at each caching node to cut down much of the network traffic.

Section 4.2 describes hash routing and other backgrounds. In section 4.3, we introduce and give details of the duplicated hash routing algorithm. We evaluate its performance and compare it with that of other algorithms. We describe simulation models of these systems in section 4.4, and examine the relationship among various parameters such as the number of caching nodes, the disk size of each node, frequency of duplication, network traffic, and hit rates in section 4.5. Finally, we conclude this paper with a summary of our results and a discussion of some open issues.

4.2 Background

A variety of technologies for distributed caching systems are being developed. We describe some of them within the scope of our study.

4.2.1 Hash Routing

Hash routing requires neither a query to a neighbor nor an exchange of an object list. Figure 4.1 depicts a basic model of hash routing, called a simple hash routing. A client sends an HTTP request to its local caching node. All the caching nodes in a system share a single hash function and are assigned a part of name space of WWW objects

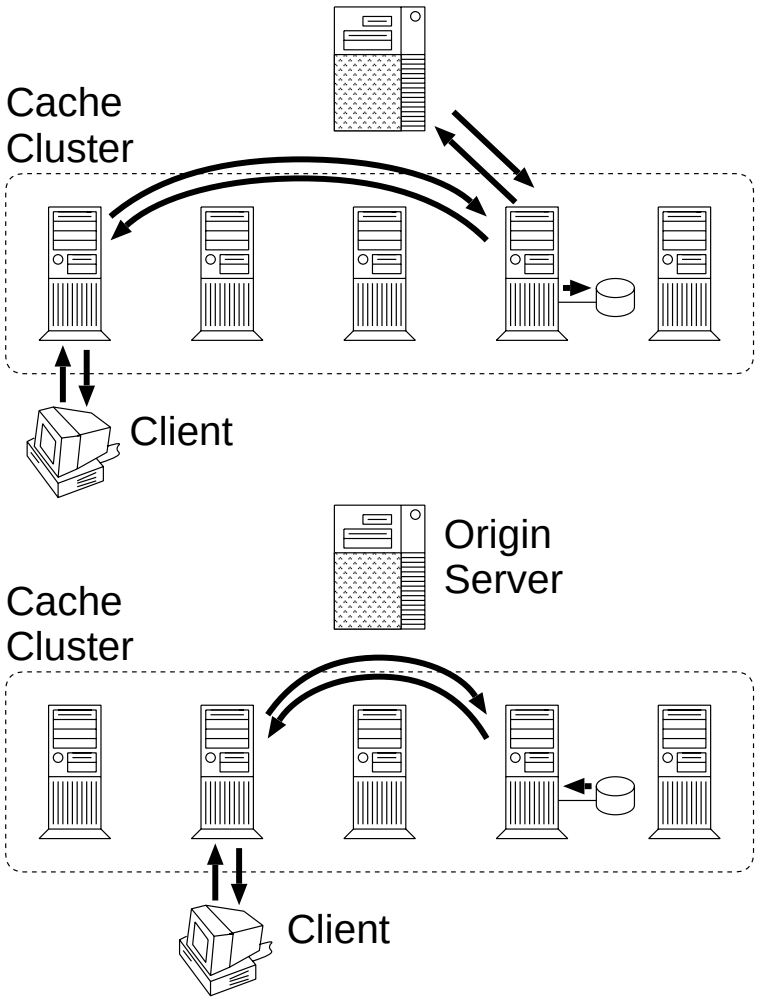

Figure 4.1: Simple hash routing (upper: cache miss, lower: cache hit)

without overlaps by the hash function. Thus the local caching node calculates a hash value of the request and forwards it to some node according to the hash value. After the object is cached at one of the caching nodes, requests for the object from any clients in the system make cache hits.

Because an object is cached at only one caching node, hash routing can achieve a higher hit rate. On the other hand, it has two drawbacks as we mentioned in section 4.1. One is its large network traffic between caching nodes. Because a client cannot retrieve objects directly from the cache of its local caching node in most cases, a large number of object transmissions between caching nodes occur in the hash routing system. Therefore hash routing requires the situations that all the caching nodes are within short distance and connected to each other by networks with high bandwidth. The other drawback is its lack of fault tolerance. Because an object is only cached at one node, all clients can suffer from errors or cache misses even when just a single caching node in the system gets failure.

4.2.2 Robust Hash Routing

There are many possible points of failure in a distributed WWW caching system. In our study, we focus on failures at caching nodes. We do not discuss other points such as networks or origin servers because a failure on a network can be counted in a failure of a caching node and a failure at an origin server has little impact on the total availability of service compared with that at a proxy server.

When a failure occurs at a caching node of a hash routing system, rebuilding its hash function is a straightforward way to recover. However, there is a weak point that the fraction of objects no longer in correct caches can be large [21]. This leads to performance degradation of the whole system during and after failure of a caching node. To overcome such weakness, *robust hashing* was proposed [4] and actually implemented [20].

With robust hashing, the URL of a requested object and the name of each sibling cache together are used to generate a hash value or score; the object is then mapped to the sibling cache with the highest score. This technique keeps all cached objects valid. However, a certain degree of performance degradation is still inevitable.

4.2.3 Proxy Auto Configuration

As a client-side technology to evade failure of a local caching node, the Proxy Auto Configuration (PAC) technology is widely used. When PAC is used, a client receives a list of proxy servers. If the local proxy is in failure, the client can bypass it and forward its requests to one of other caching proxies according to the proxy list. In our study, we assume all clients use the PAC technology.

In case of hash routing, PAC is not sufficient. For example, if a remote caching node gets failure, the objects cached at the node cannot be retrieved from the cache. All the requests for such objects result in error or cache miss depending on system implementation. In our study, we assume that such errors are avoided by forwarding the requests to the origin servers directly or to some other caching nodes.

4.3 Duplicated Hash Routing

The goal of hash routing is to optimize system cache capacity. The key idea in our algorithm, duplicated hash routing, is introducing minimum redundancy to keep its high hit rate when some of the caching nodes are in failure. Our algorithm differs from robust hash routing, mentioned in section 4.2.2, in the point that our algorithm prepares for system failure and keeps its performance during the failure period.

As illustrated in figure 4.2, an object is cached at a single node in the same way as the simple hash routing algorithm. We call this caching node a *primary* cache of this object. When a following access to the object makes a cache hit at the primary cache, it duplicates the object to another cache, which we call a *secondary* cache of the object. If the primary cache gets failure, the request for the object is forwarded to the secondary cache and makes a cache hit.

In our system, all the caching nodes share two hash functions. One is used to decide the primary cache of an object and the other is to decide the secondary cache of the object. By this cache redundancy, we can achieve robustness against failure.

Besides object duplication, we optionally enable local caching at each node. Local caching can reduce network traffic between caching nodes. Here, we discuss hit duplication and local caching in our system.

Hit Duplication With duplicated hash routing, each caching node copies its cached object to the secondary node. If the object duplication is performed frequently, the traffic between caching nodes gets increased significantly. Moreover, these duplications introduce some overlap between caches; this algorithm wastes system cache capacity. Because these copies are only for fault tolerance capability, we have to minimize the frequency of copying.

At this point, we can consider two types of solutions. One is avoiding the object duplication when the object is not expired at the secondary node. With this technique, we can reduce the network traffic caused by the object copies. However, to know whether the secondary cache has the object, some kind of querying protocol such as ICP is required, which makes the system design and implementation complicated. The other solution is taking long intervals between object duplications. If this technique can suppress the network traffic to a permissible degree without performance degradation, the querying protocol is not necessary.

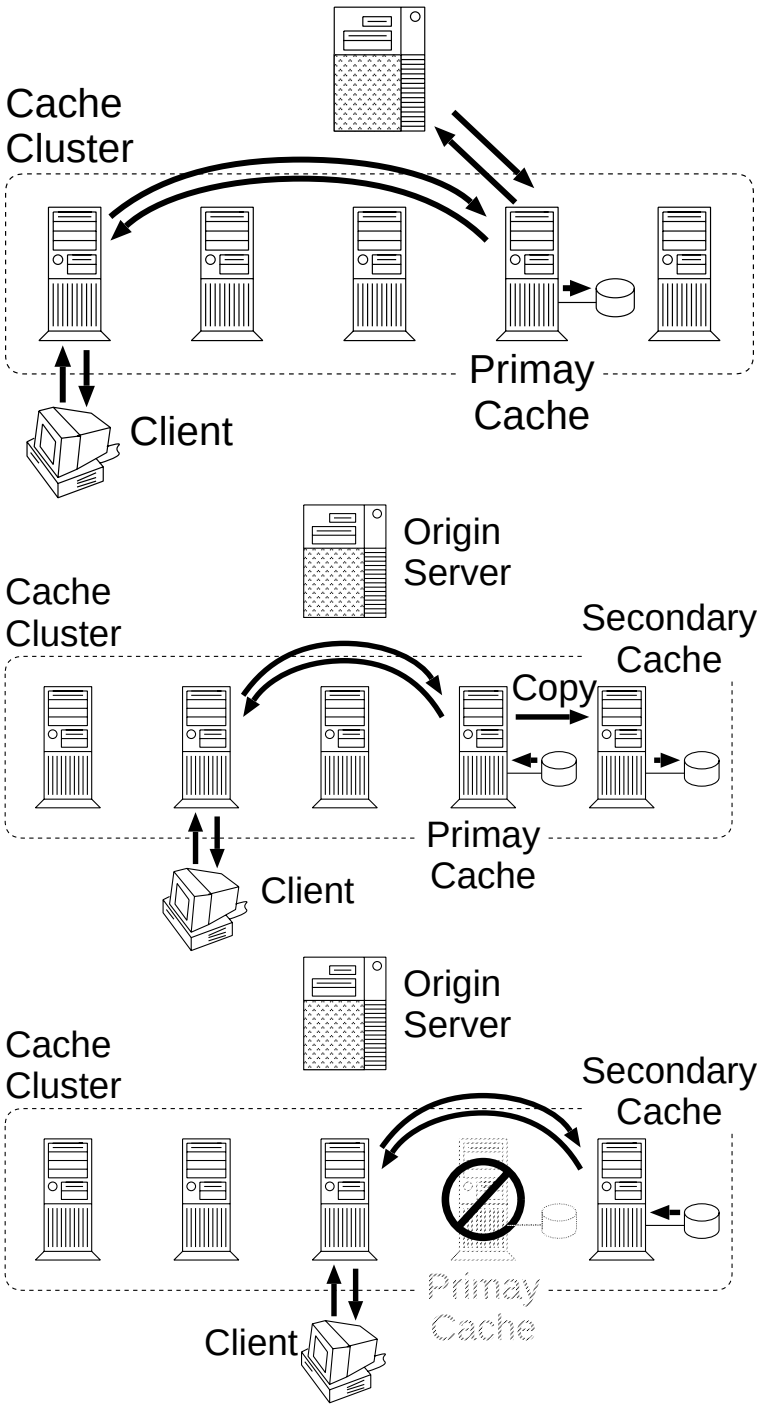

Figure 4.2: Duplicated hash routing (upper: cache miss, middle: cache hit and copy, lower: failure and cache hit at the secondary cache)

Local Caching The hash routing algorithms remove overlaps of caches among nodes and optimize the system cache capacity. In compensation for its high hit rate, network traffic among caching nodes is fairly large. Though we assume the situation that the caching nodes in the system are connected with each other by high speed networks, traffic localization in the system still has great significance for system performance such as reduction of service delay. For that reason, we can optionally allow each caching node to cache objects requested by its local clients. Of course, local caching reduces the system cache capacity. If the hit rate decreases substantially when local caching is enabled, the benefit of hash routing is spoiled. Therefore, we have to examine whether the system has adequate cache capacity before enabling local caching.

4.4 Simulation Model

We evaluate the duplicated hash routing algorithm by simulation. In this section, we describe some topics on the simulation to be considered.

4.4.1 Workload

Web accesses received at a caching proxy show distribution in conformity with Zipf's law[1] [1, 13]. More precisely, the number of requests f for an object is expressed as $f = C/r^k$, where r is the access ranking of the object and C and k are constants depending on access characteristics of the user community. However, Zipf's law cannot explain the characteristics of whole web accesses from various communities.

To simulate a distributed WWW caching system, we put a strong assumption that the total set of requests from all clients in the system follows Zipf's law. Although this assumption satisfies the requirement that the distribution of accesses received at each caching node conforms Zipf's law, the sufficient condition of this proposition is not proved yet. Even if this assumption is somewhat different from actual situations, we can get some hints on the transition of hit rates or network traffic according to the variation of several parameters of the system.

In Zipf's law, k means the scatter of requests from clients and C means the order of the total number of requests. In our simulation, we set the parameter of k at 0.66, which is reported in [22]. From the parameter k and the total number of requests, we can calculate the parameter C. In our simulation, we set C at 50,000 and generate about 35,000,000 requests.

[1] This distribution is called a "Zipf-like" distribution in [13], because it does not follow strictly the original Zipf's law.

4.4.2 Object Size

The distribution of WWW object size conforms log-normal distribution [23]. Its probability distribution function is expressed as follows.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp\left(\frac{-(\log x - \mu)^2}{2\sigma^2}\right)$$

We set the parameters of μ and σ at 7.3 and 1.7 respectively observed in [22] and get the probability distribution of object size. Using this distribution, we choose the size of each object within the limit of 100MB. Too large objects give not a small influence to cache efficiency especially when its capacity is relatively small, even though appearance of such large objects is quite rare. Moreover, download of such a huge object through WWW is unrealistic.

4.4.3 Replacement Algorithms

Dozens of object replacement algorithms for a cache are proposed, and developing better algorithms is still one of hot topics in the field of WWW caching (for example, see [24, 25, 26]). However, a better cache replacement algorithm is not the main motivation of this study, we adopt a classical LRU algorithm in our simulation, which is proved to achieve a moderate hit rate compared with other algorithms.

4.4.4 Failure Rates

System failure can be characterized by two factors, mean time between failures (MTBF) and mean time to recover (MTTR). In our simulation, we take the number of requests received by a caching node in substitution for passage of time at the caching node. In other words, our simulation is request-driven.

We fix the parameters of C and k to produce about 35,000,000 requests in total and assume that 2% of these requests meet failure at their local caching nodes[2]. We set the mean number of request receptions between failure at 100,000; each caching node recovers after a certain number of request receptions, which is determined randomly from 0 to 200,000. Under this situation, the occurrence probability of failure at each request is set at 0.02/100,000 (2×10^{-7}).

To discuss robustness against system failure, it is important to use realistic values of parameters. In our simulation model, at a caching node which receives 100,000 requests per day from its clients, MTBF is 50 days and MTTR is one day (and this node shall be repaired within two days from an occurrence of failure). We consider this situation is typical of many organizations where caching proxies are operated.

[2]However, these requests are forwarded to some other nodes and do not result in errors by the PAC technology.

4.4.5 Copy Intervals

With the duplicated hash routing algorithm, a cache hit may cause an object copy. Each caching node has its threshold of a copy interval. When a cache hit occurs, the node checks the time when it copied the object for the last time. If a longer period than the threshold passes after the last copy, the node tries to copy the object to the secondary node. A system with capability of querying an object to the secondary node checks whether the object is cached at the secondary node; it can duplicate the object only when the object is already expired.

In our simulation, the threshold of an interval between object duplications is decided by the number of requests the caching node receives from its local client. By default, object duplication can occur after d/s receptions of accesses, where d and s denote the cache size of the node and the mean size of objects, respectively. Here, we introduce a copy factor, a factor of the threshold, and examined the cases that the copy factor is 0.0, 0.25, 0.5, 1.0, 2.0, 4.0, and 8.0. The copy factor of 0.0 means that caching node checks and/or duplicates the object every time a request makes a cache hit. The copy factor of 8.0 means that a caching node duplicates an object after $8.0d/s$ receptions of requests.

4.4.6 The Number of Nodes and Cache Size

In a simulation of a WWW caching system, it is important to choose appropriate disk size against the number of requests processed by a caching node. In our study, the system receives about 35,000,000 requests as a whole. This means that total of about 300GB of data is retrieved by clients. We set the total system cache capacity at 32GB, 64GB, 96GB, 128GB, and 160GB. We can examine both situations with poor and rich cache capacity.

Cache capacity of each node is decided by this total capacity. For example, the capacity of a single cache is set at 4GB when the system capacity is 64GB and the number of nodes in the system is 16.

4.5 Results

We simulated caching systems, stand alone caching, ICP-based caching, robust hash routing and our duplicated hash routing. We discuss the performance of these algorithms in terms of hit rates, cache capacity, and network traffic. Robust hash routing and duplicated hash routing are examined in both cases with local caching enabled and disabled. In case of ICP and stand alone caching, local caching is always enabled.

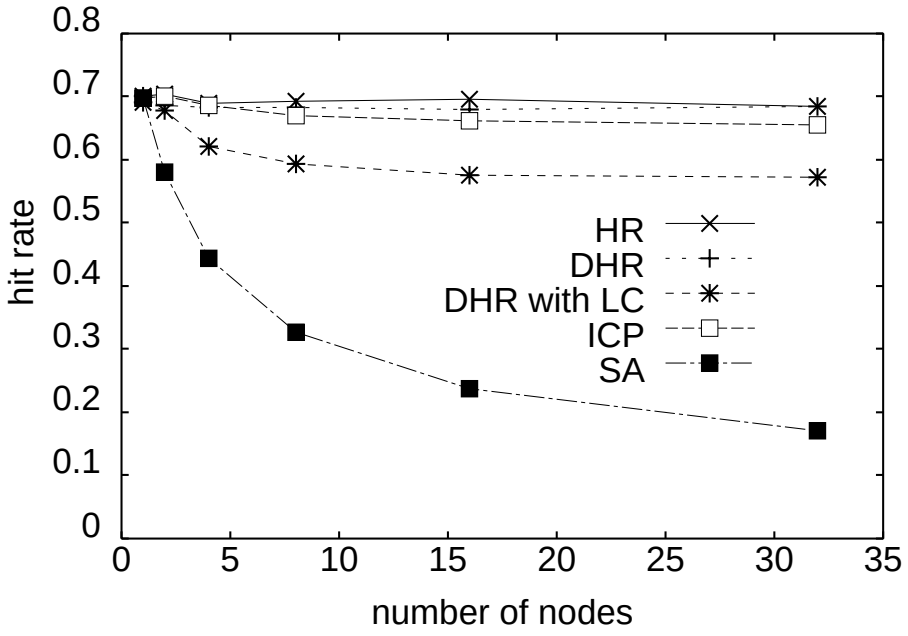

Figure 4.3: Hit rates of robust hash routing (HR), duplicated hash routing (DHR), duplicated hash routing with local caching (DHR with LC), internet cache protocol (ICP), and stand alone caching (SA) with 64GB of cache capacity

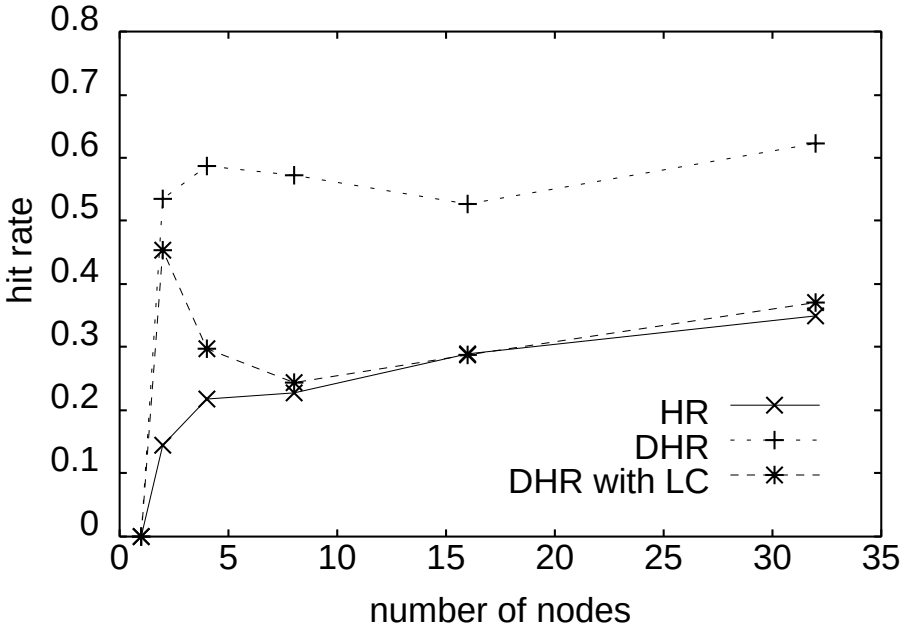

Figure 4.4: Hit rates on failure (with 64GB of cache capacity)

4.5.1 Hit Rates

Figure 4.3 depicts total hit rates with the robust hash routing algorithm, the duplicated hash routing algorithm, the internet cache protocol, and the stand alone caching. As this chart shows, distributed caching systems can achieve higher hit rates.

The algorithm that gets the highest hit rate is robust hash routing. Because overlaps of caches between nodes are removed with robust hash routing, the hit rate is kept high even if the scale of the system grows. Duplicated hash routing also attains a hit rate as high as robust hash routing does. This implies that the decrease of system cache capacity caused by cache duplications has almost no influence on the hit rate of the system. However, when we enable local caching, the hit rate decreases about 10%. This means that we cannot ignore the decrease of system cache capacity caused by local caching. Here, we have to recall the suitability of system cache capacity to the number of requests. The high hit rate of the hash routing algorithm indicates that this capacity is large enough to produce ideal results. We will discuss the cache capacity later in section 4.5.2.

Contrary to our expectation, the difference in hit rates of ICP and hash routing is small. However, this high remote hit rate is obtained at the expense of large traffic between caching nodes, as we will mention later in section 4.5.3.

In case of failure at a caching node, both robust hash routing and duplicated hash

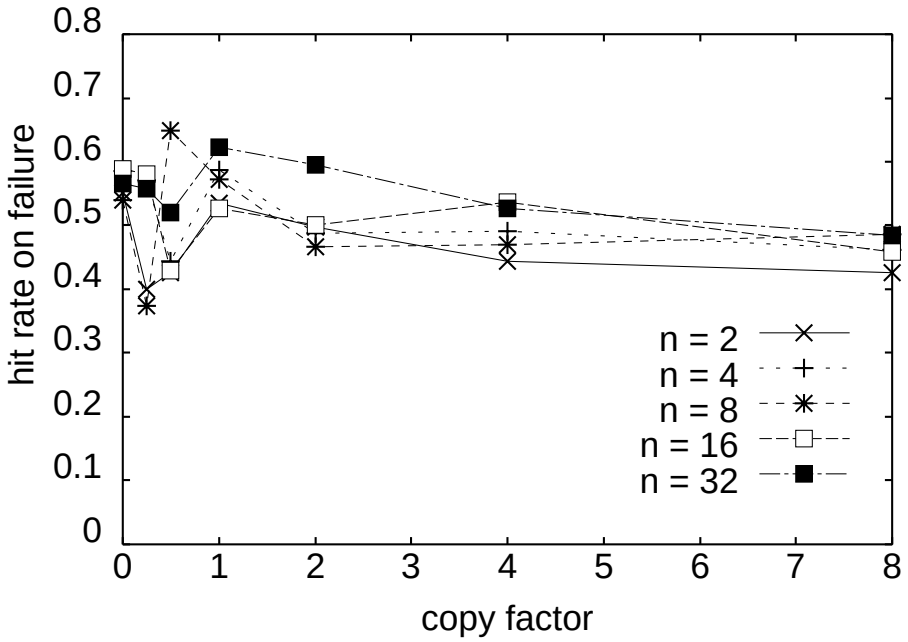

Figure 4.5: Hit rate on failure and copy factor of duplicated hash routing (n is the number of nodes and cache capacity is 64GB)

routing can continue their services to their clients. The objects which are to be cached at the failed node are cached at some other nodes decided by their scores or the secondary hash function. In contrast to robust hash routing, duplicated hash routing can prepare cached objects at secondary nodes for system failure. In other words, as for failure, robust hash routing always makes a *cold start*. We extract requests for objects which are to be cached at failed nodes and examine the hit rates of them. As figure 4.4 shows, our algorithm keeps high hit rate even in case of failure. When local caching is enabled, however, the hit rate degrades considerably. Local caching decreases system cache capacity and that gives more influence on these hit rates than on the total hit rates. We will discuss this topic later in section 4.5.2.

We also examine a relationship between hit rates in case of failure and copy factors, and the result is depicted in figure 4.5. Though we expected that the hit rate decreased rapidly as we increased the copy factor, this result indicates that our expectation is fortunately not true. The impact of the copy factor on the hit rate is turned to be fairly small.

4.5.2 Cache Capacity

In the previous section, we mentioned that local caching wasted the cache capacity of the system. This means that we can keep the high hit rate if we add extra cache capacity to the system. We simulated the systems with robust hash routing and duplicated hash routing in case with and without local caching.

Figure 4.6 shows the hit rates of two algorithms without local caching. From this result, we can see that 64GB of system cache capacity is enough to this request sequence. When we set the capacity at 32GB, the hit rate gets evidently lower than the others. However, the hit rates with more than 64GB of cache capacity are almost the same.

Figure 4.7 represents the cases we enable local caching. With low cache capacity, local caching makes hit rates low. However, with larger cache capacity, we can prevent the decline of hit rates caused by local caching. From the result, we can see that almost the same hit rate as the case without local caching can be achieved with 128GB of cache capacity in both hash routing systems and duplicated hash routing systems.

Figure 4.8 shows a correlation between the hit rate in case of failure and the total cache capacity using our algorithm with local caching enabled. As we mentioned in section 4.5.1, this hit rate degrades when local caching is enabled. However, this graph indicates that a fairly high hit rate can be achieved with larger cache capacity.

4.5.3 Network Traffic

Figure 4.9 shows the network traffic between caching nodes. In this result, we count neither the local traffic between clients and their local caching nodes nor the traffic

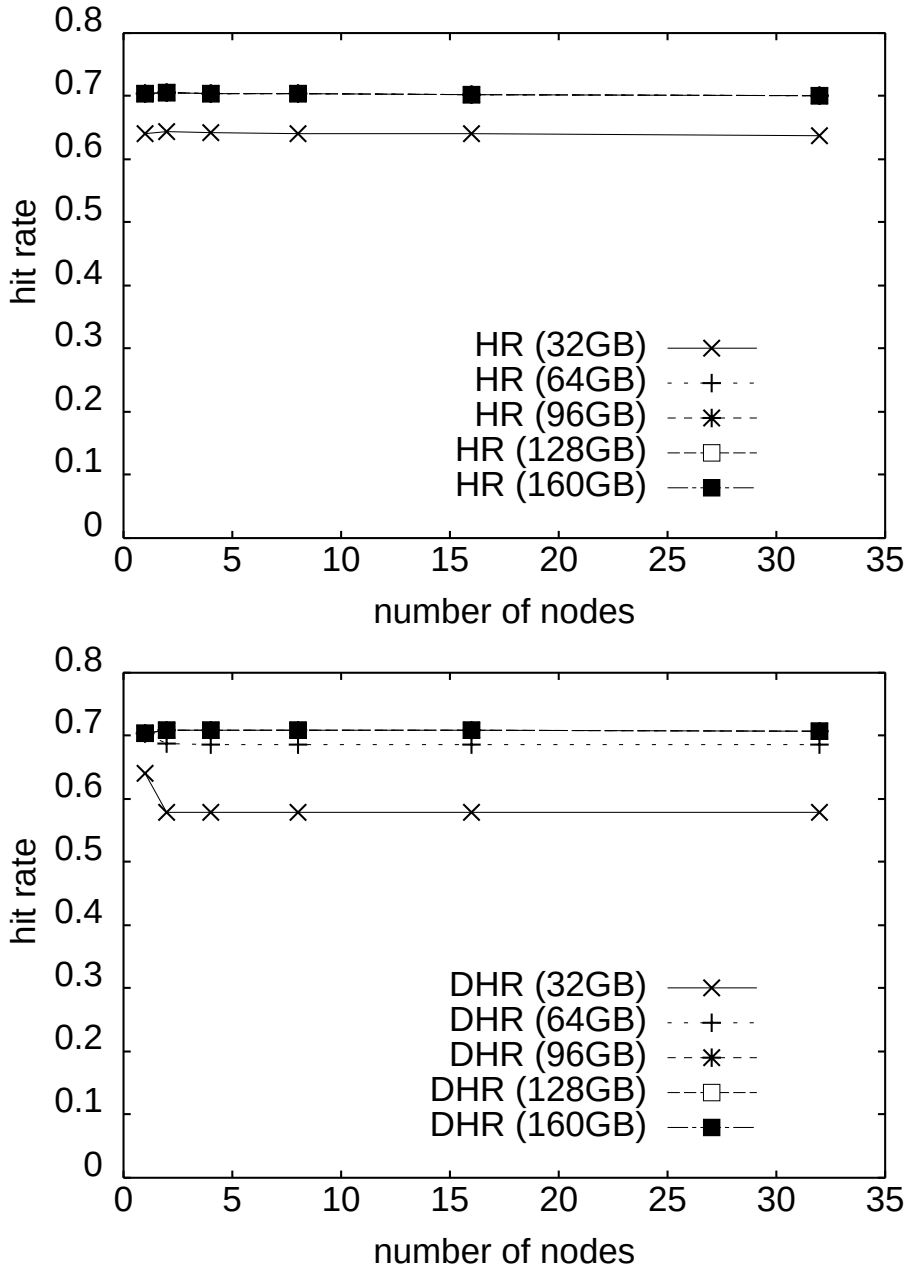

Figure 4.6: Hit rates with local caching disabled (upper: robust hash routing, lower: duplicated hash routing)

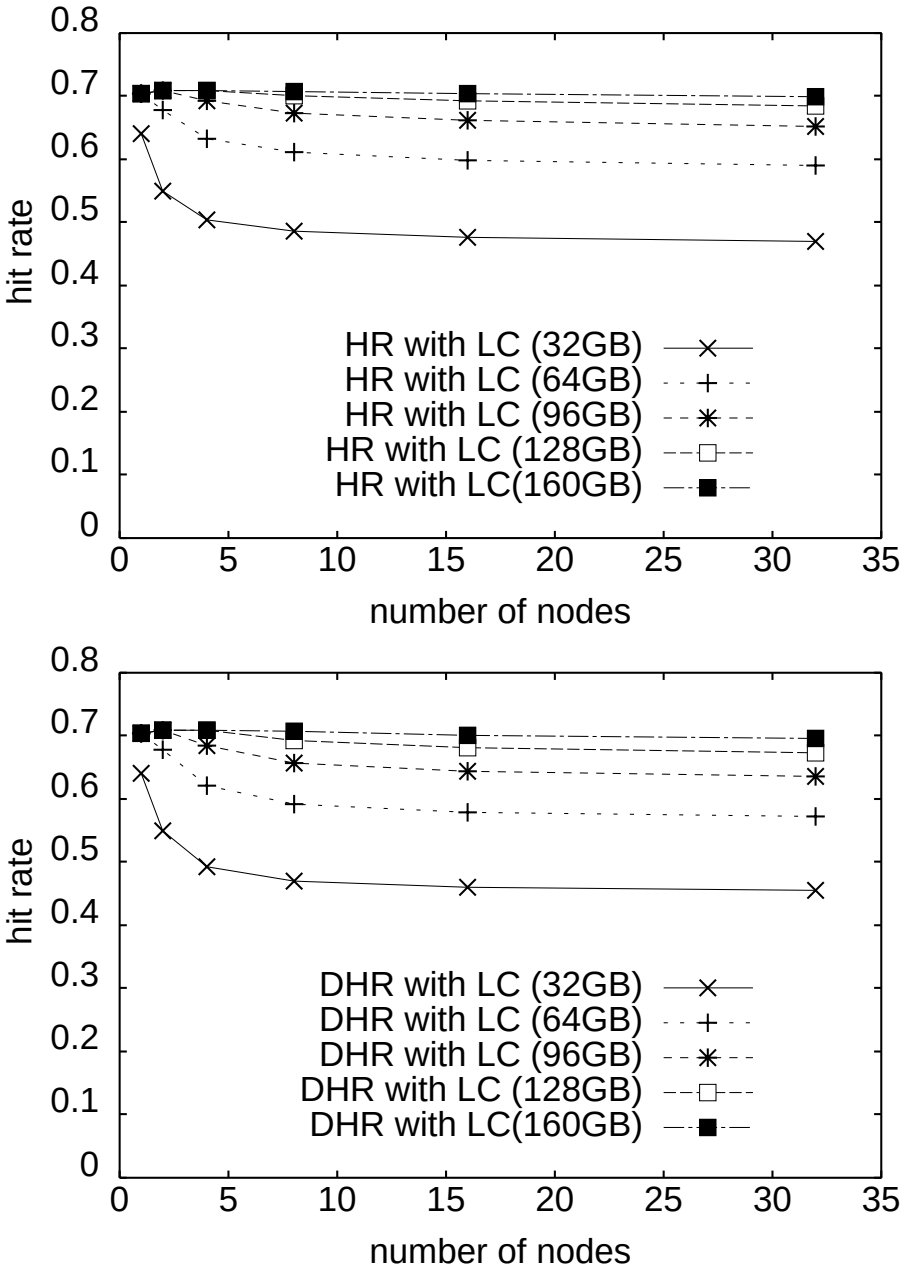

Figure 4.7: Hit rates with local caching enabled (upper: robust hash routing, lower: duplicated hash routing)

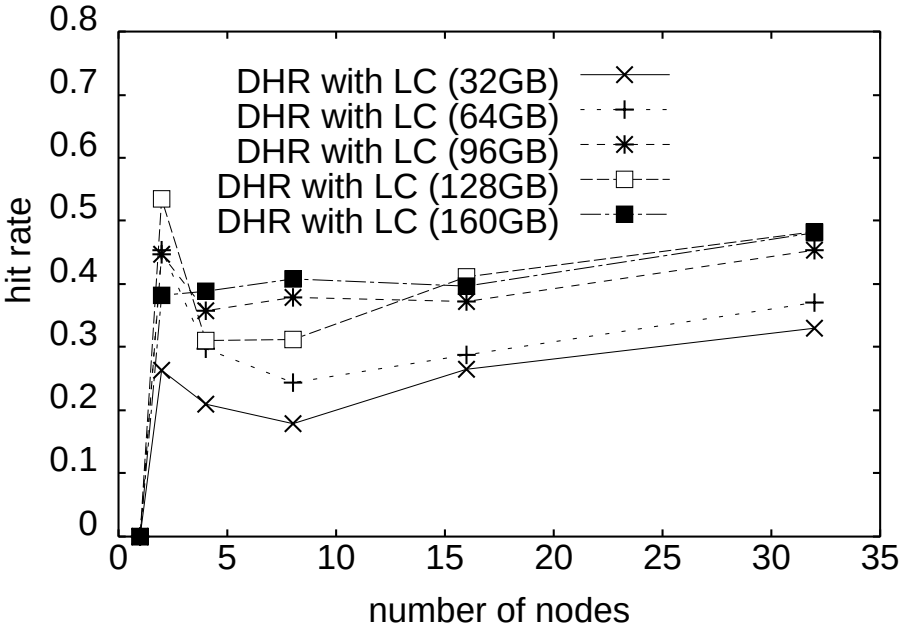

Figure 4.8: Hit rates of duplicated hash routing on failure with local caching

between caching nodes and origin servers, because we focus on intra-system traffic here.

When the number of caching nodes is small, ICP requires relatively small network traffic between nodes. However, ICP cannot adapt to a large scale system, because it uses broadcast to send queries to all nodes. Even if each query is small in its size, the total network traffic between caching nodes gets explosively large as the system scale grows up. On the other hand, the increase rates of network traffic with robust hash routing and duplicated hash routing are fairly low. The difference in the traffic between these two algorithms is caused by object duplication that is a feature of duplicated hash routing. To make the network traffic low, it is the best way to transmit an object to the secondary node only when the object is not cached by the node. However, it requires a querying protocol and takes some communication cost between caching nodes. Instead of this querying technique, we take an approach of employing a large copy factor. In our simulation, we set the copy factor of duplicated hash routing at 4.0 and it makes the network traffic between caching nodes fairly low. We will revisit this topic later.

When we enable local caching in a duplicated hash routing system, the increase rate of network traffic gets slightly increased. However, compared with the duplicated hash routing algorithm without local caching, it significantly reduces the network traffic especially in case with a small number of caching nodes.

As we mentioned before, we can decrease the traffic between nodes by increasing the copy factor. Figure 4.10 shows the relation between the copy factor and the network

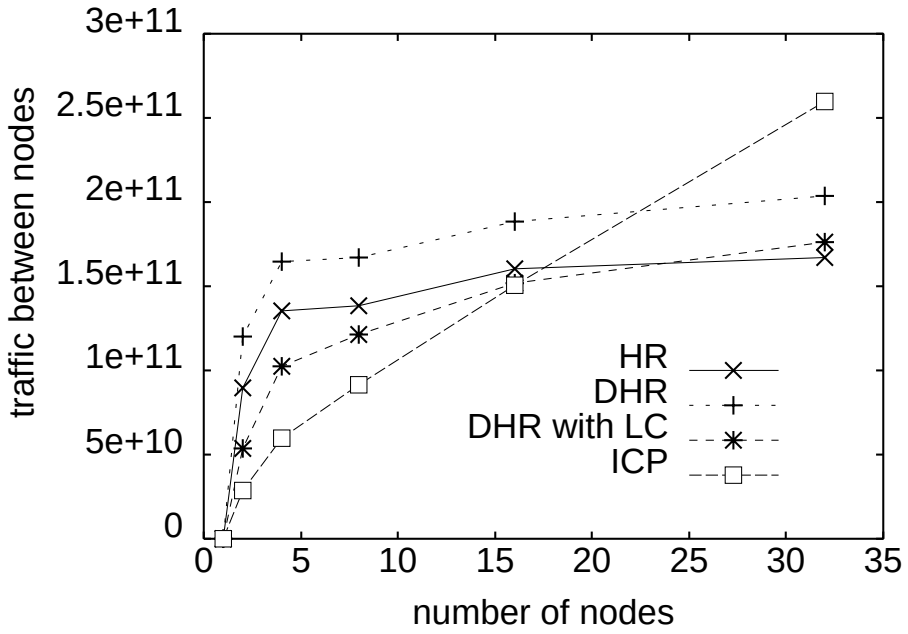

Figure 4.9: Network traffic between nodes with hash routing, duplicated hash routing, duplicated hash routing with local caching, and ICP (with 64GB of cache capacity)

Table 4.1: Summary of hash routing, duplicated hash routing, duplicated hash routing with local caching, ICP, and stand alone caching

	total hit rate	hit rate on failure	network traffic between nodes	scalability
HR	high	medium	medium	medium
DHR	high	high	medium – large (improvable)	medium
DHR with LC	medium (improvable)	medium (improvable)	medium (improvable)	high
ICP	high	high	depends on the number of nodes	low
SA	low	nil	nil	–

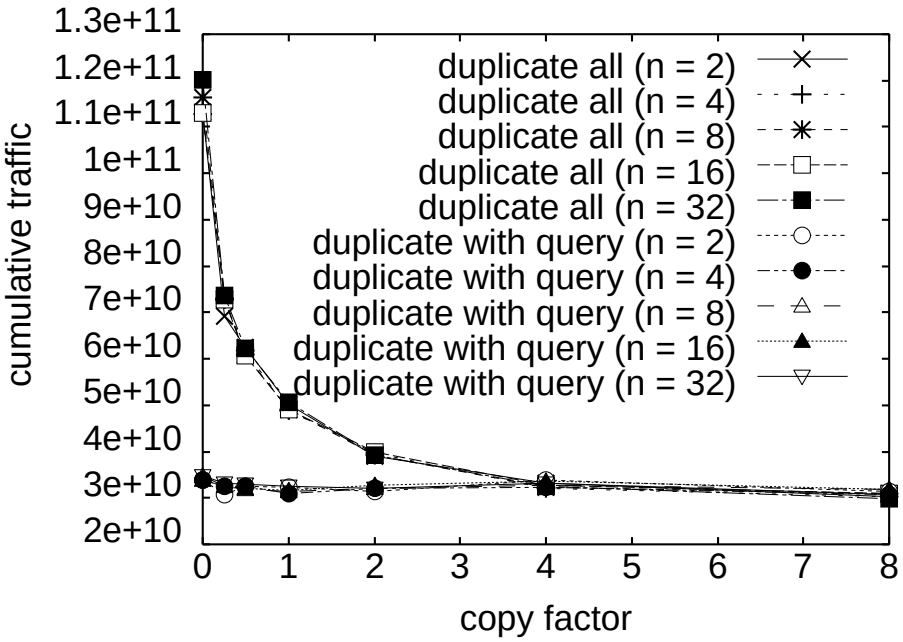

Figure 4.10: Network traffic and copy factor with duplicated hash routing (n is the number of nodes and cache capacity is 64GB)

traffic between caching nodes. This graph shows two patterns: one is a case that each node always duplicates an object without a query after an interval based on its copy factor and the other is a case with a query. In the former case, unnecessary object duplications raise the network traffic when the copy factor is small. In the latter case, we put the communication cost of sending a query to the secondary node is the same as that of an ICP query. From the results, the difference between the two patterns is neglectable if we set the copy factor larger than 4.0.

Considering the fact that the copy factor does not affect on the hit rate so much, as we mentioned in section 4.5.1, it is unnecessary to send queries to secondary caches if we set the copy factor at more than 4.0. If we avoid such queries, we can keep the system design simple and its implementation easy.

4.5.4 Summary

We summarize the features such as total hit rates, hit rates on failure, network traffic between caching nodes, and system scalability of each algorithm in table 4.1. Our extension to hash routing gains durability against system failure at a relatively small expense of increased network traffic. Moreover, this expense can be reduced by local caching and a large copy factor. ICP also achieves both high durability and high hit rates. However, its scalability is highly restricted by its large increase rate of network traffic.

4.6 Concluding Remarks

WWW becomes an indispensable service on the Internet and the fault tolerance of systems which provide the WWW service has great importance. We have proposed the duplicated hash routing algorithm for a distributed WWW caching system, which has robustness against failure. Our simulation results suggest that our algorithm can achieve a high hit rate even when some caching nodes are in failure. In addition, optionally we can enable local caching at both robust hash routing system and our duplicated hash routing system. If we enable local caching, the traffic between caching nodes is reduced, though the cache capacity of the system gets lower. Therefore, the total hit rate also gets lower when local caching is enabled. This phenomenon can be prevented as we add extra cache capacity to the system. If we increase the cache capacity, we can both keep the high hit rate achieved by robust hash routing and reduce network traffic between caching nodes substantially. In duplicated hash routing system, intervals between object duplications have great importance. With a relatively long copy interval, we can omit the object querying mechanism from the system and design the system to duplicate objects always after the interval because the increase of network traffic is small enough.

Chapter 5

Conclusion

Various cache technologies are applied to improve the quality of WWW services. It can reduce WWW service latency, traffic on the Internet, load of WWW servers, etc. This dissertation focuses on the traffic on distributed WWW caching systems. In Chapter 2, we reached the conclusion that we have to minimize the number of siblings, especially, external siblings from a viewpoint of the border gateway. ICP does not decrease any traffic on the border gateway. Therefore, if the performance bottleneck of WWW access is on the low bandwidth of the connection between the local area network and the Internet, the distributed WWW caching system scattered over the Internet with ICP does not help any traffic reduction on the bottleneck. In such a situation, improving the local hit rate is more important. We have a lot of configuration parameters such as cache size in memory or disks and cache expiration strategies according to object types. By tuning these parameters, we can possibly obtain several percent increase of the local hit rate, which is more effective than that of the remote hit rate.

From a viewpoint of WAN, the traffic analysis presents a different picture. As we mentioned in Chapter 3, ICP can decrease the WWW traffic on the Internet though the traffic on the border gateway increases. In our analysis, we concluded that the remote hit rate over 4 or 5 percent is required for the overhead of ICP traffic to exceed the amount of reduced HTTP traffic by ICP. This is not a realistic situation if we use a well-tuned proxy server. As a special case of ICP utilization, if we use ICP in the situation that the distance between the proxy server and the sibling proxy server is very small, the required remote hit rate becomes quite small. In other words, ICP is suitable for clustering proxy servers in an organizational network composed by several LANs. Because the interconnection bandwidth of LANs is relatively high compared with that of WANs in many cases, the problem of border gateway bottleneck is small.

From those two chapters, we achieved two objectives of this dissertation, i.e., quantitative evaluation of the traffic on a distributed caching system and distinct guideline to decide the proper and improper situation for adoption of ICP. Although ICP is a simple and easy protocol to use, its effect on the reduction of WWW traffic is very

difficult to attain and requires quite deep consideration.

Beside ICP, hash routing is an algorithm that does not exchange meta information among caching proxies and removes object duplication among proxies. Thus, the total logical cache size of the hash routing system can be larger than that of ICP. On the other hand, hash routing has the problem of low tolerance to failure. Because each caching node has to handle the requests from all clients in the system, failure of a single node inevitably has influence on all the clients. In Chapter 4, we proposed duplicate hash routing that introduced slight cache redundancy and achieved the third objective, that is, the improvements of fault tolerance and efficiency of hash routing.

In the traffic analysis of the caching system using ICP, some works still remain to be done both theoretically and practically as an open issue. Theoretically, we need to analyze other algorithms such as cache digest [27], Crisp cache [28] and Summary cache [29]. The essential drawback of ICP is the lack of scalability. As the scale of the system grows, i.e., the number of siblings increases, the number of ICP queries increases and most of them result in waste of network resources. Those algorithms listed above avoid sending queries to each siblings per local cache miss and solve this problem. They exchange a list of cached objects periodically in a highly efficient way. The analyses of these algorithms are one of our future works.

Practically, we need to verify our model through experiments such as packet monitoring, which can show the actual behavior of cache systems in more details. The important point is that our theoretical model gives sufficient insight into the drawbacks of ICP. However, in our analysis, we lack discussion about actual network traffic. For example, our model takes no account of packet retransmission. Moreover, the TCP algorithms implemented in operating systems differ slightly from each other. There is a limit of analysis accuracy in the theoretical method.

We also need to verify our algorithm, duplicated hash routing, with actual implementation. Although the duplicated hash routing algorithm achieves high hit rate and reduction of network traffic between caching nodes, it requires complicated implementation. Because one of the major goals of WWW caching system is reducing the service delay, the implementation should be efficient. Such implemention of duplicated hash routing is our future work.

Chapter 6

Future Works

Currently, the distributed cache technology is applied in various situations and the way the technology is applied has been changing every day. The infrastructure of information distribution is becoming richer and getting more diversity. Here, I can enumerate several trends that will have much influence on the next generation information distribution systems and give some open issues about the trends.

6.1 More Network Bandwidth

The Internet gets more network bandwidth. There is an opinion that the cache technology is no longer necessary if the network bandwidth becomes large enough. However, the larger network bandwidth does not mean the extinction of performance bottleneck in the network services. It means that the performance bottleneck has moved from the past narrow backbone networks to the servers that host popular web sites. In this situation, the cache technology can be applied to remove this bottleneck. To apply the cache technology to a high-speed network environment, it is necessary to implement a high-speed cache server. Expecially, bandwidth in the order of Gbps is available but no single cache server can handle such large bandwidth. Such implementation of a cache server requires various deep considerations about memory management, connection/session management, thread/process management, cache expiration strategy, etc. We need not only the consideration about software but about hardware. There are many problems in the current computer hardware when we develop such high performance servers. For example, the internal bus speed of server hosts is not enough in many cases. Especially, the low bandwidth of peripheral bus such as PCI is one of performance bottleneck of such high speed network services. In addition, we have to pay attention to the interaction between the software and the hardware. The software that manages the peripheral devices, that is, an operating system is also the performance bottleneck. Development of an operating system for high speed I/O is my future

research theme.

6.2 Multimedia Streams

Multimedia stream data becomes more popular and many people download stream data through the Internet. There are several difficulties in the application of the cache technology to multimedia services. Because the size of multimedia data is large and its download time is long, the traditional passive cache technology cannot be applied. In this situation, how to distribute the multimedia data in advance to intermediate cache servers on the Internet is important. This technology, called active cache compared with the traditional passive cache, is a hot research topic. If only the popular contents are concerned, the solution is simple. Because the number of popular contents is small enough compared to that of all the contents on the Internet, it is relatively easy to distribute them to the cache servers all over the world. However, if we apply the cache technology to a video on-demand (VoD) system that serves various sorts of stream data, there are a lot of problems. For example, a large number of non-popular stream data should be distributed all over the world. In addition, the server has to handle the service requests randomly. In this situation, we need more cooperative system that scattered on the Internet to relieve these hardships.

6.3 Peer-to-Peer

Today, peer-to-peer (P2P) information systems get reality. As an extension of the traditional client server communication model, a lot of peer-to-peer (P2P) information systems are proposed and some of them are actually operated. With the P2P communication model, the distinction between a client and a server does not make sense because the P2P system is a truly distributed system where all the nodes work together both as a client and as a server. The major problem of this communication model, however, is its high consumption of resources such as computation power and network bandwidth. Because each host relays a request to the next hop hosts blindly with this P2P model, the request data can be duplicated many times on the Internet. These requests put high load on many hosts. To solve this problem, meta data cache technology can be applied on this model. For example, the meta data of popular contents should be duplicated near clients. Because the communication cost of exchanging meta data is far lower than that of exchanging the data itself, meta data of various contents can be kept at many hosts.

Bibliography

[1] Ari Luotonen and Kevin Altis. World-Wide Web proxies. In *Proceedings of the 1st International WWW Conference*, Geneva, Switzerland, May 1994. http://www1.cern.ch/PapersWWW94/luotonen.ps.

[2] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical Internet object cache. Technical Report 95-611, Computer Science Department, University of Southern California, March 1995. ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/HarvestCache.ps.Z.

[3] Internet cache protocol (ICP), version 2. RFC 2186, September 1997.

[4] Keith W. Ross. Hash-Routing for Collections of Shared Web Caches. *IEEE Network*, pages 37–44, November/December 1997.

[5] Squid internet object cache. http://squid.nlanr.net/Squid/.

[6] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. Caching proxies: limitations and potentials. In *Proceedings of the 4th International WWW Conference*, Boston, MA, December 1995. http://www.w3.org/pub/Conferences/WWW4/Papers/155/.

[7] Bradley M. Duska, David Marwood, and Michael J. Freeley. The measured access characteristics of World-Wide-Web client proxy caches. In *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, Monterey, CA, December 1997. http://www.cs.ubc.ca/spider/marwood/Projects/SPA/wwwap.ps.gz.

[8] D. J. Barnes and N. G. Smith. An analysis of world-wide web proxy cache performance and its application to the modelling and simulation of network traffic. In *Proceedings of the 4th International Conference on Telecommunication Systems Modeling and Analysis*, page 9, March 1996.

[9] Kenichi Yoshida. Distributed cache layout based on access patterns (in Japanese). In *Proceedings of Internet Conference '96*, 1996.

[10] Masahiko Nabe, Ken-ichi Baba, Masayuki Murata, and Hideo Miyahara. Analysis and modeling of WWW traffic for designing internet access network (in Japanese). *The Transactions of IEICE*, J80-B-I(6):428–437, 1997.

[11] Duane Wessels and K Claffy. ICP and the Squid Web cache. *IEEE Journal on Selected Areas in Communication*, 16(3):345–357, April 1998. http://www.ircache.net/ wessels/Papers/icp-squid.ps.gz.

[12] Steven Glassman. A caching relay for the World-Wide Web. In *Proceedings of the 1st International WWW Conference*, Geneva, Switzerland, May 1994. http://www1.cern.ch/PapersWWW94/steveg.ps.

[13] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of the INFOCOM '99 conference*, March 1999. http://www.cs.wisc.edu/ cao/papers/zipf-like.ps.gz.

[14] Michael Rabinovich, Jeff Chase, and Syam Gadde. Not all hits are created equal: Cooperative proxy caching over a wide-area network. In *Proceedings of the Third International WWW Caching Workshop*, June 1998. http://www.cs.duke.edu/ari/cisi/crisp/neighbors.ps.gz.

[15] Renu Tewari, Michael Dahlin, Harrick Vin, and John Kay. Beyond hierarchies: Design considerations for distributed caching on the Internet. Technical Report TR98-0, University of Texas, February 1998. http://www.cs.utexas.EDU/users/dahlin/papers/tr98-04.ps.

[16] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, third edition edition, 1996.

[17] Application of Internet cache protocol (ICP), version 2. RFC 2187, September 1997.

[18] David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins, and Yoav Yerushalmi. Web Caching with Consistent Hashing. In *Proceedings of the 8th International World Wide Web Conference*, Toronto, Canada, May 1999.

[19] Cisco Systems Inc. *Cisco Cache Engine, Version 2.0*.

[20] Vinod Valloppillil and Keith W. Ross. Cache array routing protocol v1.0. Internet draft, February 1998. http://www.ietf.org/internet-drafts/draft-vinod-carp-v1-03.txt.

[21] D. G. Thaler and C. V. Ravishankar. Using Name-Based Mappings to Increase Hit Rates. *IEEE/ACM Transactions on Networking*, 6(1), February 1998.

[22] Ken-ichi Chinen. *Studies on Effective Methods of Providing and Retrieving Information in the Internet*. PhD thesis, Nara Institute of Science and Technology, February 1998.

[23] Masahiko Nabe, Ken-ichi Baba, Masayuki Murata, and Hideo Miyahara. Analysis and Synthesis of World-Wide-Web Traffic (Japanese). Technical Report SSE96-90, IN96-74, CS96-98 (1996-09), IEICE, 1996.

[24] Hyokyung Bahn, Sam H. Noh, Kern Koh, and Sang Lyul Min. Using Full Reference History for Efficient Document Replacement in Web Caches. In *Proceedings of USENIX Symposium on Internet Technology and Systems*, Boulder, Colorado, USA, October 1999. USENIX Association.

[25] Madhukar R. Korupolu and Michael Dahlin. Coordinated Placement and Replacement for Large-Scale Distributed Caches. In *Proceedings of the IEEE Workshop on Internet Applications*, San Jose, CA , USA, July 1999. IEEE.

[26] B. Krishnamurthy and C. Wills. Proxy Cache Coherency and Replacement - Towards a More Complete Picture. In *Proceedings of the 19th International Conference on Distributed Computing Systems*, Austin, Texas, USA, June 1999. IEEE.

[27] Alex Rousskov and Duane Wessels. Cache digests. In *Proceedings of the 3rd International WWW Caching Workshop*, June 1998. http://www-sor.inria.fr/mirrors/wcw98/31/ rousskov@nlanr.net.ps.

[28] Syam Gadde, Jeff Chase, and Michael Rabinovich. A Taste of Crispy Squid. In *Proceedings of Workshop on Internet Server Performance*, Madison, Wisconsin, USA, June 1998.

[29] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *Proceedings of ACM SIGCOMM'98 Conference*, Vancouver, British Columbia, Canada, September 1998.

[30] Yusaku Hasegawa. Research on an effective mechanism to discovery path MTU (in Japanese). Master's thesis, Nara Institute of Science and Technology, 1997.

[31] Bruce A. Mah. An emperical model of http network traffic. In *Proceedings of the IEEE Infocom '97 Conference*, Kobe, Japan, April 1997. IEEE.

[32] Alex Rousskov and Valery Soloviev. On peformance of caching proxies. In *Proceedings of the Joint International Conference on Measurement and Modeling of*

Computer Systems (SIGMETRICS '98/PERFORMANCE '98), pages 272–273, June 1998.

Appendix A

Estimation of the Number of Packets per Request

We estimate the number of packets per HTTP request and per ICP query exchanged among clients, proxies, and origin servers. In our estimation, some results of our log analysis on several caching proxy servers are used. In this appendix, we show how the number of packets is estimated on HTTP and ICP.

A.1 Estimation of the Number of Packets per HTTP Request

HTTP uses TCP as its transport layer protocol. The number of packets exchanged by a client and a server is modeled as Figure A.1. At first, a 3-way handshake is done to open a TCP connection. Next, a GET request is sent from the client to the server and its ACK packet is returned from the server to the client. Since most URLs have the size under 100 bytes and the average size of URLs is about 50 bytes, we can assume that the GET request is sent in one IP packet. Next, the requested WWW object is returned from the server to the client. The object is fragmented into segments whose size is the maximum segment size (MSS) of the connection and each segment is put into a single IP packet with IP headers. It is known that the MSS is 1460 bytes in most cases [30]. In our estimation, we assume the MSS is 1460 bytes. Considering the distribution of object size derived from our log analysis and algorithm of TCP, we estimate the average number of exchanged packets (including ACK packets not shown in the figure) for object transfer is 9.35. After object transfer from the server to the client is completed, FIN packets and their ACK packets are exchanged to close the connection.

The above is the typical model of packet exchange at an HTTP transaction. We estimate the average number of packets per HTTP request is 18.35 altogether.

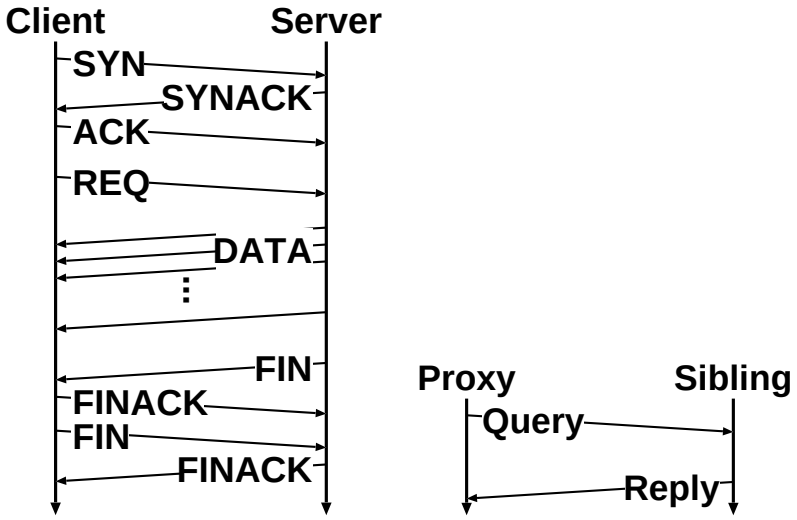

Figure A.1: Models of an HTTP request (left) and a ICP query (right)

A.2 Estimation of the Number of Packets per ICP Query

ICP uses UDP as its transport layer protocol. The number of packets exchanged by a caching proxy and a sibling is depicted in Figure A.1. The proxy send a ICP query to the sibling. The sibling receives the ICP query, searches the requested object in its cache and returns the result.

In the ICP message, a 20-byte header and a payload that contains the URL of the requested object are placed. Because the average size of URLs is observed to be about 50 bytes, we assume each message is on just one IP packet. After all, 2 packets are exchanged.

Appendix B

Characteristics of WWW Traffic

We made statistical analysis on the log data generated by Squid operated at our institute for a month in May 1998. Many studies on the characteristics of the WWW traffic have been done so far, and many of them reported similar results (e.g., [31, 32]). Here we show the details of our analysis as a help for your understandings of our communication cost evaluation discussed in this dissertation.

B.1 Object Size

Our Squid caching proxy system is in operation on SUN Microsystems Ultra Enterprise 3000 server (4 CPU of Ultra Sparc 168MHz, 256MB of memory). Our Squid is configured to use 60MB of memory and 2GB of hard disks for its operation. We also set up that the Squid refers 8 sibling proxies on the Internet, however, this sibling configuration is only for the purpose to evaluate the ICP performance. More than three million HTTP requests are processed for a month in this Squid server. Here, we divided the results into 4 categories described in Section 3.5.1. Table B.1 shows the summary of our statistical analysis. The distribution of the object size for each category is depicted in Figure B.1, B.2, B.3, and B.4.

Table B.1: Summary of WWW accesses at our institute (May 1998)

Result	number of requests	rate	average object size
LOCAL HIT	1132961	0.34274	4606 bytes
LOCAL MISS	2172607	0.65726	8790 bytes
REMOTE HIT	215376	0.06516	6958 bytes
ALL MISS	1957231	0.59210	8993 bytes

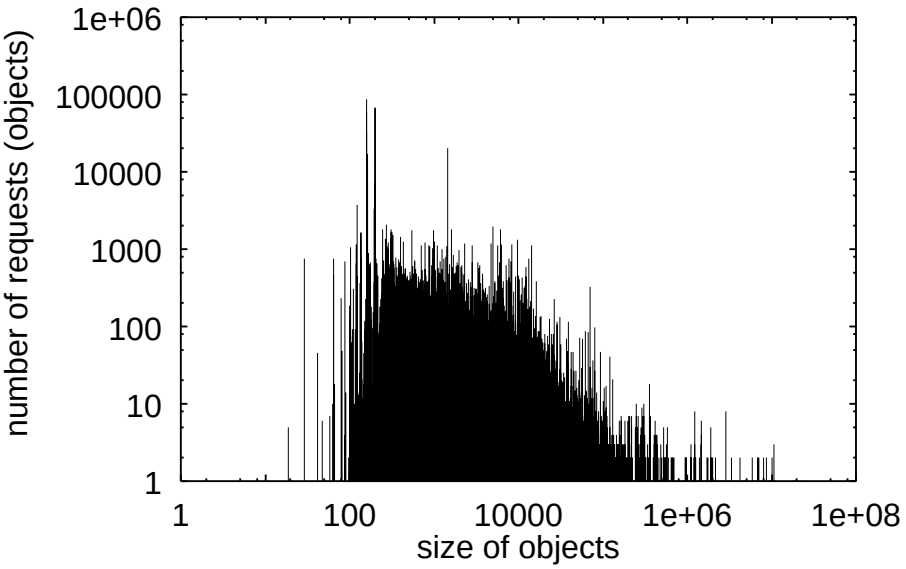

Figure B.1: Distribution of object size (LOCAL HIT, May 1998)

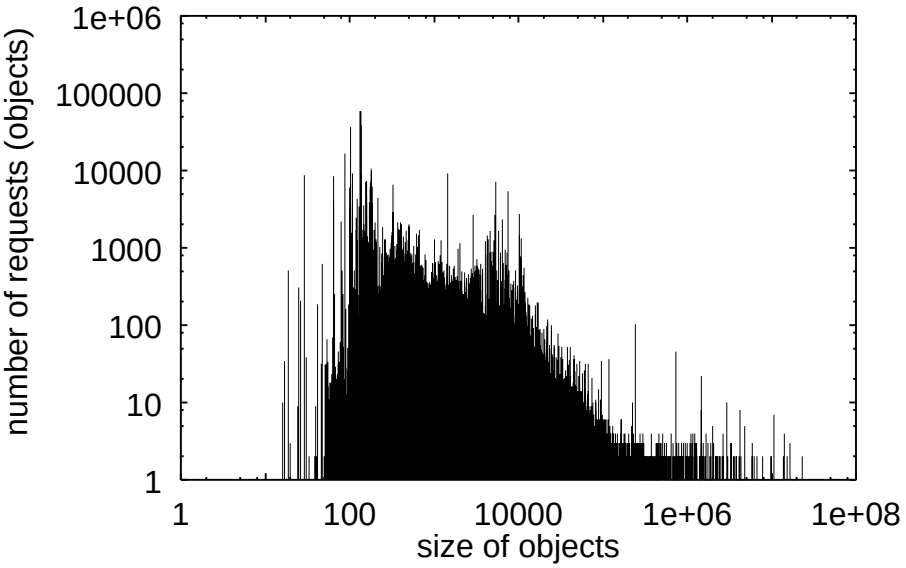

Figure B.2: Distribution of object size (LOCAL MISS, May 1998)

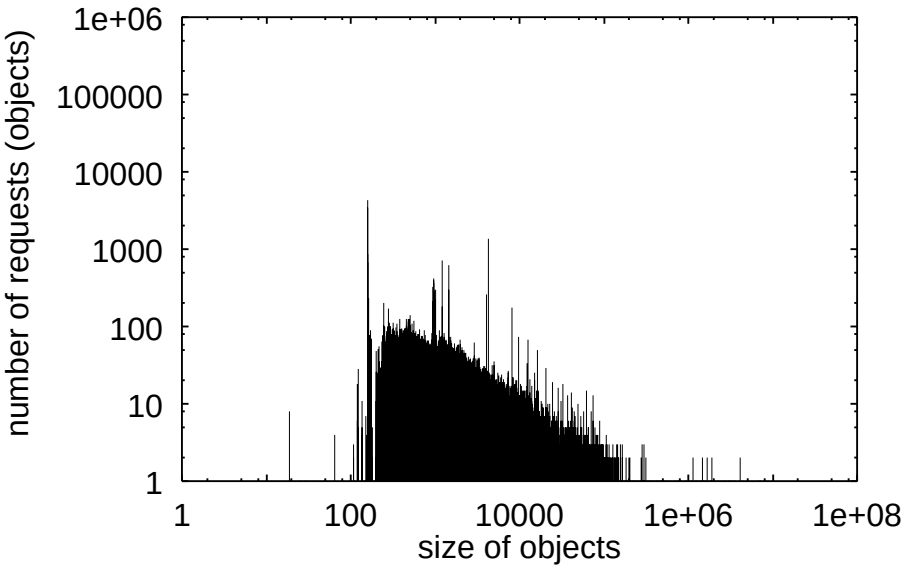

Figure B.3: Distribution of object size (REMOTE HIT, May 1998)

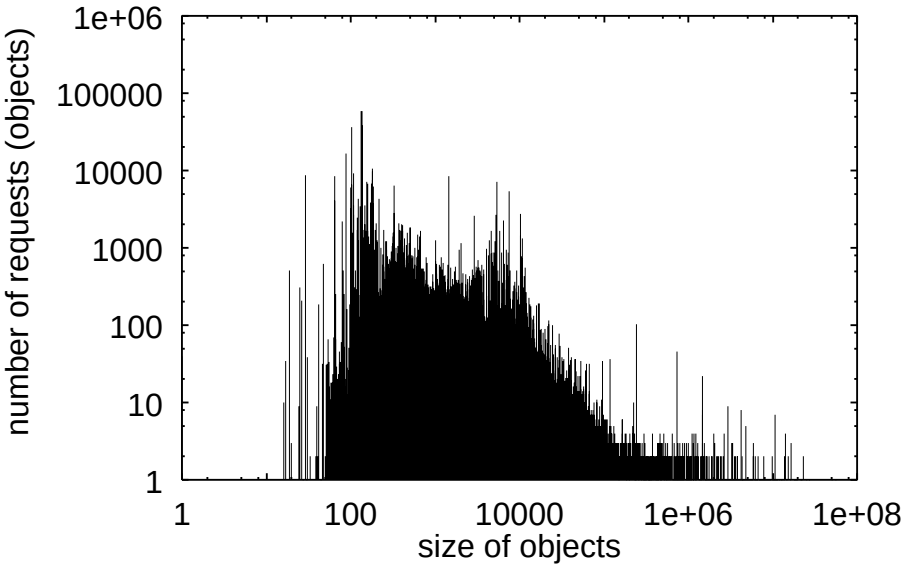

Figure B.4: Distribution of object size (ALL MISS, May 1998)

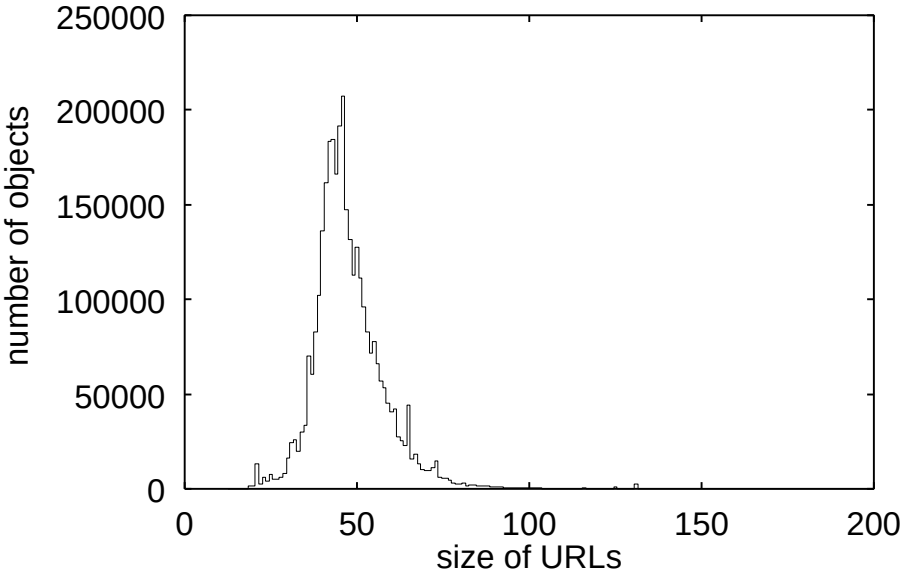

Figure B.5: Distribution of size of URLs

In order to estimate the average size of ICP messages, we investigate the average size of URLs in HTTP request. An ICP message consists of a 20-byte header and a URL in an ASCII text. The distribution of the size of URLs is shown in Figure B.5. The average size of the URL we observed is about 50 bytes. Consequently, about 70 bytes can be given as an estimation of the average size of ICP messages.

B.2 Number of Packets on HTTP

Based on the actual observations on the number of the packets through our operation of Squid, we plot its distribution in Figure B.6, B.7, B.8, and B.9. Note that this distribution has interesting characteristics. The log of the number of requests in which requested objects are divided into P packets is almost proportional to $\log P$.

We estimated the number of packets exchanged on HTTP with two assumptions: (1) the maximum segment size (MSS) of TCP is 1460 bytes, and (2) any packets for the retransmission due to network congestion are ignored. In the TCP connection establishment phase, three packets are transferred between a server and a client. Then, the client sends an HTTP request in a single packet and the server sends an ACK packet back to the client. Here, we define B as the size of the WWW object the client requests. The object is divided and stored in packets. For each data packet, a single ACK packet is sent from the client to the server. With our analysis on the implementation of the HTTP, the number of the ACK packets can be supposed as described in Table B.2.

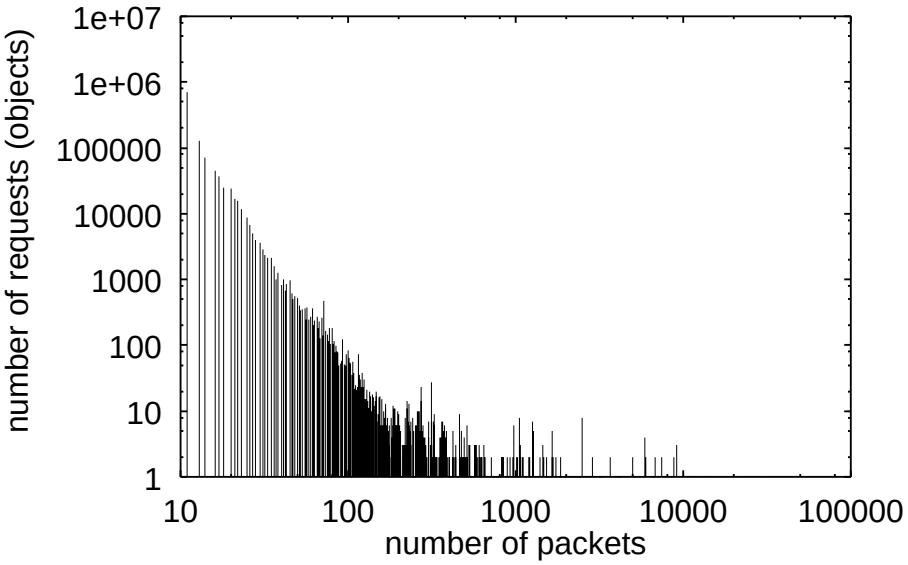

Figure B.6: Distribution of number of packets (LOCAL HIT, May 1998)

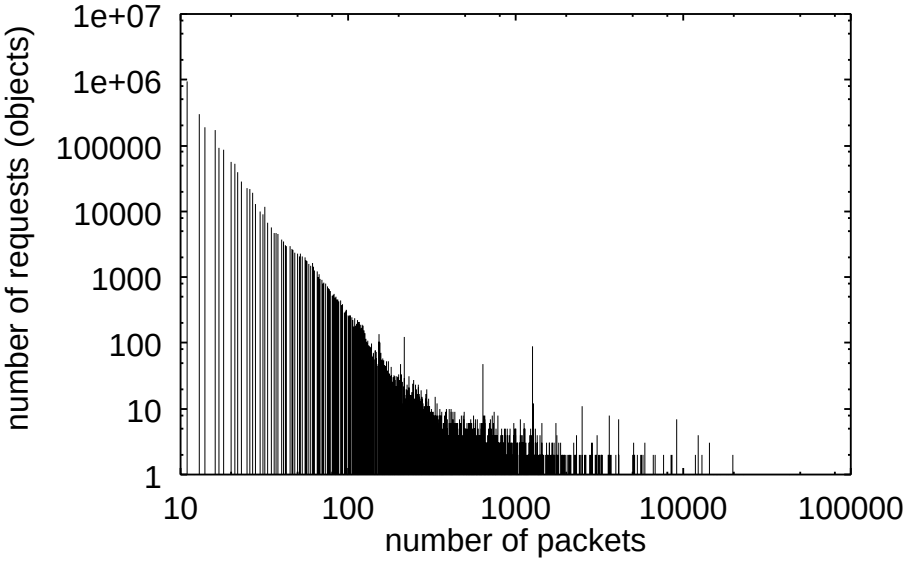

Figure B.7: Distribution of number of packets (LOCAL MISS, May 1998)

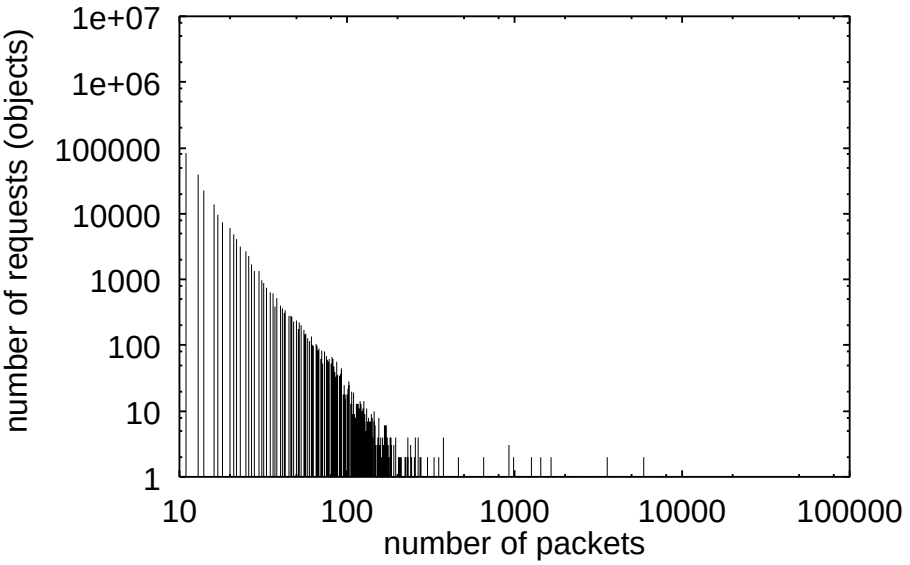

Figure B.8: Distribution of number of packets (REMOTE HIT, May 1998)

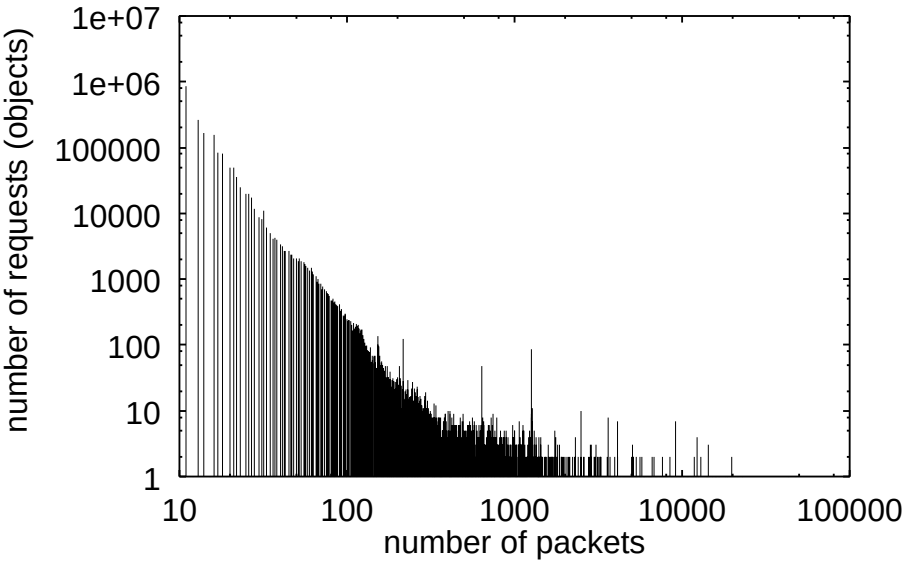

Figure B.9: Distribution of number of packets (ALL MISS, May 1998)

Table B.2: Number of ACK packets

T_b	Number of ACK Packets
0 - 1460	1
1261 - 4380	2
4381 - 8760	3
over 8760	an additional ACK per 4 data packets

Consequently, we can formulate the number of all the packets exchanged between the client and the server as Equation 3.1 in Section 3.5.2.