2001

# Advance web requests : reducing latency while utilizing resources more effectively via instructor initiated prefetching

Muammer Akcay
*Lehigh University*

Akçay, Muammer

# Advance Web Requests: Reducing Latency While Utilizing Resources More Effectively...

January 2002

# ADVANCE WEB REQUESTS: REDUCING LATENCY WHILE UTILIZING RESOURCES MORE EFFECTIVELY VIA INSTRUCTOR INITIATED PREFETCHING

by

Muammer Akçay

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science and Engineering

**Lehigh University**

**2001**

This thesis is accepted in partial fulfillment of the requirements for the degree of Master of Science.

10/8/01

(Date)

Prof. Terrance E. Boult
Department of CSE
Dissertation Advisor and Chairperson

Prof. M. Schulte
Department of CSE

# Acknowledgments

It has been a tremendous privilege to work with Professor Terrance Boult. It remains a mystery how anyone can be so thoroughly knowledgeable in such a wide variety of fields. His boundless energy, enthusiasm, optimism, and timely words will not be forgotten. I would like to thank Terry for his quick understanding and unique approach to problems, and resolving more effectively. Many thanks to him for his intellectual generosity and scientific skills he gave me to explore many of the ideas in this thesis, the keen advice he offered when needed, and generous financial support.

I would like to thank Aydin Yeniay for his enormous support and guidance on time.

Many thanks to Ali S. Erkan for his tremendous support and unique cognizant guidance in helping me find my destiny. His interaction with people, motivation, infinite and timeless efforts will not be forgotten.

I would like to thank the AT&T Foundation for supporting some of this work. I would like to thank Professor Leroy Tuscher and his students. I would like to thank the people at Broughal Middle School, especially Dawn Bothwell for implementing and collecting experimental work.

The influence of my fellow VAST Lab researchers, especially Sezai Sablak, Jason W. Kim, and Peter G. Lewis, in this work is greatly appreciated. I also wish to thank all my colleagues and friends at Lehigh University.

Although some universities in Turkey are not known very well like Lehigh, I cannot acknowledgment enough the educational institutions in Turkey especially Anadolu and Osmangazi Universities in Eskişehir. Various people and factors have touched me many ways to reach this moment of my life.

I would like to thank many friends from all over the world especially Connecticut, California, Massachusetts, New York, New Jersey, Pennsylvania, and Europe. These people have not been forgotten during my study and whole life.

I apologize to my parents for causing any unforgettable or incurable problems throughout of my life. I am ever grateful to my parents and my family for supporting, encouraging, and guiding me during these uncountable years of higher education, sacrificing and allowing me to go so far from home.

# Table of Contents

# List of Tables

# List of Figures

# Abstract

*One word is too much; many words are too little to explain*

Since the Internet has become popular, the amount of data on the World Wide Web (WWW or simply the web) has been increasing rapidly. Because the demand for Internet access has grown much faster than technological developments, increasing web performance becomes an important issue. Some important research and implementations have been done on web caching to improve web performance during the last half decade. Although conventional web caching uses previously accessed web data (objects), recent studies and implementations are moving in a new direction to predict future web object requests. The instructor initiated web caching technique predicts web objects for a group of clients which have almost the same interest. The instructor initiated web caching allows a reduction in latency, and utilizes web resources (e.g., network bandwidth, and server capacity) usage by predicting client requests.

Client waiting time, latency, shows satisfaction of web interaction from the user's point of view. Latency consists of web server and data transmission path implicitly. Latency and other performance metrics can mislead understanding web performance. Definition of web performance and comparison with other studies are important when representing client satisfaction.

This work explores a novel approach on building interest groups through web caching hierarchy and distributing predicted objects. Generalizing prediction techniques and object (content) dissemination through the cache hierarchies via multicast or semicast for a group of clients are new research directions to achieve better web performance.

Since conventional web requests are done by on demand user requests, having advance web requests will have tremendous opportunity to utilize web resources more effectively. In addition, client requests can be done in advance by requesting web objects for a time interval. Advance web requests have a high potential to achieve better web performance by consuming web resources effectively.

The result of this research will lead to a collaboration of clients, network resources, and web servers and improve coordination between proxy cache servers. In addition, this study leads improved information dissemination through the Internet.

**Keywords:** Web caching, instructor initiated prefetching, prediction and prefetching for a group of clients, content dissemination, advance web request, web performance, latency.

# Chapter 1

# Introduction

*Nothing is forever.*

*A small key opens big doors.*

*An open door invites callers.*

In this chapter, a brief introduction to the World Wide Web is given. Then, the problem being explored is introduced. Necessity of this work follows later. Contributions from this work will be summarized. Finally, organization of the thesis will follow.

## 1.1 Introduction to the Web

Since the cost of personal computers has become affordable, their performance has been continuously improving, and they are good enough to satisfy personal necessities, fewer people can resist using computers. Many applications are developed and available for people to use computers. When each individual person wants to communicate via computers, they should be connected in some fashion. Various computer network topologies for communication have been developed such as ring, star, mesh, and tree, etc., depending upon the necessity of the application. A communication of computers through the networks has also been developed, which is known as the Internet. Many data communication protocols have been developed such as Token Ring, User Datagram Protocol (UDP) [Pos80] Transmission Control Protocol (TCP) [rfc81b], Internet Protocol (IP) [rfc81a], etc. Beyond the Internet by using communication protocols, many application protocols have been developed, such as file transfer protocol (FTP), hypertext transfer protocol (HTTP) [BLFF96, Fea97], etc.

The World Wide Web, the Web, was first introduced at CERN for sharing research documents in nuclear physics. The Web is a very large distributed digital information distribution environment. After it was introduced in 1991, it has grown to encompass data dissemination of diverse information resources such as research information, personal information, digital online libraries, product and company information, any kind of publicly available information, and private information. Since many people take the Web seriously, it has been growing so fast and popular in many fields and applications.

The World Wide Web allows clients and servers to share digitally recorded information through the Internet. The Web has become gradually more popular during the last decade. A web server, which is a host computer that serves publicly available digital information to clients; and a web client server, which makes client requests to the web servers via the Internet; have been developed. Many web servers and web client servers (browsers) are available to the clients. The web server and the web client exchange messages over the Internet to move a web document between the web server and the web client. When a group of web clients close to each other request the same web document for a particular time interval, most of the client requests will generate the same network traffic on the path where the requests and documents travel between the web server and the web client. If there are enough resources to satisfy all client requests, every client will be happy. When web resources are limited (usually the case), some clients have to wait to be served.

One quick solution to satisfy all clients is having enough web resources such as fast web servers, fast and broad network connections, and fast client servers. Spending more money on web servers, client servers and network connections is not always a good solution. It is not possible to upgrade network infrastructure to pursue broad communication bandwidth immediately even if the cost is not an issue because of necessity of time and supplies. A client may not have a fast web server unless a client is eager to pay the cost such as dedicated service. Sometimes, lack of sufficient server capacity may cause long client waiting time even though there is enough communication bandwidth between clients and servers. When network traffic is not balanced well, having fast servers and broad network bandwidth are not enough to achieve an optimal solution. Even if having enough web resources, there can be long delays retrieving web objects due to lack of resource coordination.

Since technological improvements to access the web objects through Internet are not fast and broad enough to satisfy broad network bandwidth demand, the web performance is becoming an important issue. The number of web pages has doubled every 4 months for last couple of years. The amount of digital information has been increasing rapidly as well. When the Web was first introduced, the amount of transferred data was small compared to recent web activities. Today, web pages are sophisticated and consist of many files (sometimes large). Although communication bandwidth between web servers and clients has been increased, client web requests have grown much faster. Some web applications (e.g., multimedia, binary file distribution, etc.) require many documents and a large amount of data to transfer through the Internet. In addition, some web clients still have very limited communication bandwidth and have to share with others. Finally, web performance is becoming a bottleneck to achieve higher client satisfaction.

Since web information is growing very fast and data communication technologies via the Internet have not improved fast enough, importance of web performance still remains. Another possible solution is to combine a group of web clients that have similar web requests. Instead of making individual web requests, sharing the information among other clients will reduce web server and network traffic load. It does not only improve web performance (by reducing latency) but improves web resource utilization as well. Since network bandwidth between clients and web servers is limited, or even constant, sharing web documents plays a significant role in high web performance [NSS99]. It is shown that web caching is the best solution to achieve better web performance [WC98].

Finally, people are getting familiar with computers and the Web. When they interact with the Web more often, their expectation, accuracy, and waiting time are getting less tolerable. Therefore, web performance becomes an important issue. Accurate web objects need to be delivered to the clients as soon as possible via the Internet. To reach higher web performance, novel object delivery mechanisms (including data structure, data transfer method, data transfer medium, and effective web resource utilization) should be designed and implemented carefully. Many research projects are focused on improving web performance.

In the following section, the problem being explored is introduced.

## 1.2 Nature of the Problem

The web servers are the source of web documents (objects). The web servers run continuously and wait for web requests from the Internet. The web user or client uses an interface, called a web browser, to make a web request and see returned web objects. When a client makes a web request, the client server contacts to the original web server through the Internet. The web server processes the request and returns appropriate documents as soon as possible usually in a first come first served basis. When the returned documents are reached at the client side, the web browser shows the returned documents to the client. This completes a client request in the traditional way. It is also called direct web request.

The client makes a web request at any time. Incoming client requests to the web server can be modeled with some probability density function. Since client requests occur at random in time, one cannot determine when they happen. The web servers are always up and running to serve client requests. The web documents may not be available at the web server because of problems such as updating data, overloaded server capacity, unauthorized client request, etc. Thus, not all client requests can be accomplish when they need.

Sometimes web documents are lost between the web server and the client. Since there is no dedicated object travel path between the client and the web server, the client request may not be completed quickly and directly. When too many users try to consume available communication network bandwidth, web resources have to be shared among all of the users.

Because of limited web resources, web performance may not be satisfactory, especially during the peak web request times. When available communication network bandwidth has to be shared by others and web servers are highly loaded during the peak web request times, waiting time of the client will be longer. Web performance from the user's point of view can be defined by client's waiting time to retrieve the requested documents. When the number of users goes higher, available network bandwidth per user will be reduced if every one of the users has equal opportunity and rights to use web resources. Since client requests are made at any time, available network bandwidth for anytime varies. As a result, the user perceived time to retrieve requested documents increases. Thus, web performance from the user's point of view will be bad and unsatisfactory.

When the number of clients for a group of users at a particular Local Area Network (LAN) goes higher, their expected waiting time will be longer to retrieve web documents. Because latency has a direct impact on web performance, having the least client waiting time leads to better web performance and eventually higher client satisfaction for the Web. Because client satisfaction consists of both web server and network effectiveness of the

Figure 1.1: Instructor initiated prefetching (web caching) model

Web, reducing object retrieval time from the user's point of view as web performance is the main goal. In short, the purpose of this work is reducing waiting time of clients while utilizing web resources more effectively.

**Conventional Solutions:**

1.Web performance may not be better even though having enough web resources (upgrading web server and data transfer medium capacity) as explained later in web performance chapter.

2.Web caching which keeps frequently requested web objects in a place close to the clients in case of future requests of those stored objects is another solution to achieve better web performance. Previously requested fresh web objects are returned from the cache to accomplish client requests. When clients have limited communication bandwidth to the Internet, latency of the first access time still remains the same as whether or not employing web caching.

**Proposed Solution:**

Some of the web request patterns will highly overlap for a time interval when a group of clients works under the guidance of a leader (instructor). The clients retrieve the previously stored objects from the cache except when the objects are requested the first time. Although some prediction methods bring (prefetch) near future web objects by using previous client and object access patterns while the client request is processing, web requests for a group of clients working under the guidance of a leader can be predicted by the leader (instructor). Those predicted web objects can be brought before requested (prefetched) while web resources are lightly loaded. Since prefetched objects are brought to the cache while utilizing web resources, web server response and data transfer through the Internet can achieve maximum efficiency. The instructor initiated prefetching is shown in Figure 1.1 and explained later in detail.

Making web requests by prefetching objects while using web resources more effectively is the goal to achieving higher web performance from the client's point of view.

A group of clients requests a group of objects for a time interval while those clients work under the guidance of a leader. Making advance web requests by prefetching for the same interest groups and retrieving those predicted objects while web resources are consumed lightly are main contributions from this work. Building interest groups via cache hierarchy and distribution of objects will lead to novel content (object) distribution mechanism by the Web.

Making web requests in advance and retrieving while using web resources more effectively via prefetching will be a new avenue to reach higher web performance and client satisfaction on the Web through the Internet. Object distribution for interest groups in the cache hierarchy via prefetching are new directions of content dissemination through the Internet.

## 1.3 Distinction From Other Studies

1.Web requests of a group of users with the same interest are highly overlapped during a time interval. Sharing objects for a group of clients via web caching does not only reduce repetitive web resource consumption to retrieve the same objects but also improves web performance from the client's point of view, especially if those objects are requested for many clients.

2.When clients who take the same class have a scheduled web activity during a time interval, some of their web requests can be predicted. The instructor knows search topics, preferred sites, and interested objects of those clients. When they have a limited network bandwidth to the Internet, their waiting time will be longer even if their requests goes through the proxy server placed in their local network.

3.Prefetching for a group of users will reduce latency. Prefetched objects have a strong probability of being requested while the clients work with a leader.

4.The proposed solution makes prediction for a time interval by the instructor. Other prefetching methods are based on previous information (request access pattern of clients or objects) to predict very near future client requests. The instructor determines prefetched objects.

5.Although broad communication bandwidth in local area network (LAN) among clients and cache servers is available, network and server resources are varied between client and web servers through the Internet. Web resources should be consumed effectively to get higher web performance. Prefetching for a group of clients does not only reduce waiting time of clients but also consumes web resources more effectively.

6.When network resource consumption is well distributed, waiting time of clients will be reduced because of networking issues, e.g. congestion, overloaded links, etc. Network traffic throughout the day, weeks, or months varies a lot. Since requests are done at any time, availability of web resources varies. As a result, the client perceived waiting time for a web request goes higher while most clients are using limited connection network bandwidth to the Internet. Making some web requests in advance while distributing resource consumption in time not only reduces client waiting time but achieves better web resource utilization as well.

7.Content delivery mechanism to the interest groups via cache hierarchy is one of the novelties from this work. Predicting and distributing objects to the same interest groups while utilizing resources effectively will lead to better web performance and resource utilization.

8.Advance web requests to utilize web resources while finding best prefetching and distributing by time are contributions of better resource utilization and web performance. Requesting and keeping a group of objects for a time interval is one of the novelties from this work as well.

9.Traditionally, web requests are made and returned on client demand. The web resources may not be enough to satisfy each client request. Thus, web resources to retrieve objects have to be shared with others.

10.Advance web requests are made any time before the client request is made. Some requests can be combined and retrieved utilizing web resources.

11.The objects need to be available for a time interval. The cache maintenance can be done while the cache server is lightly loaded although conventional methods make space when it is necessary. Some of the objects may not be requested for a long time. Therefore, there is no need to keep those objects in the local cache. Fewer objects in the cache make faster cache processing (client service, object search, or object retrieval time, etc.)

12.The current web methodologies allow any request, any client, any object, any Internet connection, or any web server, anything to accomplish client request. On the other hand, some clients, client requests, objects, part of the Internet, web servers, or simply some web resources can be used for a scheduled time interval.

## 1.4 Contributions

Since the network has to be shared among clients and servers, one may wait especially during the highly loaded working times. Our purpose is to reduce the waiting time of clients while utilizing network resources more effectively. Web caching is one of the solutions to improve web performance. Predicting some future web requests for a group of clients will have potential to improve web performance. Making web requests in advance and having them available for a particular time interval is a new avenue on this field. Making a web requests in time have the advantage of effective usage of web resources. The client will make a request even if the object is not available at that time, but will be available in the future.

As a conclusion, web performance (latency) has some benefits:

1.Most frequently requested objects, which are shared by many clients, are kept in the cache, which is located close to the clients.

2.Intelligent prefetching mechanism proposed from this work for the same interest groups (clients are working under guidance of a leader) has better web performance because of effective web resource utilization.

3.Web objects are not requested forever. The instructor initiated prefetched objects are retrieved for a time interval. Thus, better cache maintenance and resource utilization will have better web performance.

4.The instructor, not necessarily based on previous request (client or object) characteristic, predicts prefetched objects.

5.Prefetching is done while utilizing web resources effectively (request time and coordination of resources to get fast data transfer).

6.This prefetching does not slow down current object retrieval since prefetching is done during the light web resource usage times unlike other prefetching techniques.

7.This work is the kernel of advance content delivery mechanism with a time interval.

8.Distributed web resource usage with advance web requests (schedule object retrieval times, optimize resource demands, and etc.) will lead to better resource utilization of exchange information.

## 1.5 Overview of Thesis

In this section, organization of the thesis will be given.

**Chapter 2** explains problem being explored in the beginning. Web performance is explained briefly. Then, motivation of this work to solve this problem is explored. Contributions of this work are summarized. Finally, judgment of this work is discussed.

**Chapter 3** summarizes overall web caching issues. Then, object prediction techniques (push and prefetching) via web caching are surveyed. In addition, comparisons of our approach with previous studies are given. Outcomes from this work are discussed briefly.

**Chapter 4** introduces the instructor initiated prefetching. Suggested model and implementation details of the instructor initiated prefetching are explored in detail. Then, experimental results are discussed. Comparison of the instructor initiated prefetching with other techniques and contributions from this part is explored. This chapter discusses future work and conclusions.

**Chapter 5** is devoted to web performance (latency). Object request states to accomplish a web request and latency factors are defined. Then, waiting time of client requests (latency) for different cases are derived and discussed. Finally, contributions from this part and future work conclude this chapter.

**Chapter 6** summarizes contributions from this work follow. Finally, conclusion remarks of this work are given.

# Chapter 2

# Background

*A man is as wise as his head, not his years.*

*See with your mind; hear with your heart.*

In this chapter, the problem being explored is described in detail. The meaning of web performance is introduced briefly. Then, motivation of this work is explored. Our approach and distinction of this work from other studies are introduced. Novelties from this work will be given shortly. Finally, how this work should be judged as engineering, scientific method, theory, and philosophy will be given briefly.

## 2.1 Introduction: Problem Statement

After the Web became popular, web performance from the client's point of view (latency) became important. Because web objects are sophisticated and consists of many documents (sometimes large amounts of digital information) while limited web resources are available to use, web object retrieval time (client waiting time, or latency) is a direct

Figure 2.1: Object retrieval diagram.

indication of web performance. It includes both web server performance and communication medium performance implicitly. When the communication network bandwidth between clients and web servers is limited and narrow, waiting time of client (latency) goes even higher. However, web server load and network resource usage are not balanced throughout a day or week. Current web requests are on a demand basis. When client requests reach to the server, they are replied to as soon as possible. When a group of users in the same local area network have the same web object requests for a time interval, each of the web requests has to go to the web server and retrieve the appropriate objects. Web caching which keeps frequently requested web objects for future client requests is located usually close to the clients. Client web requests go to the web caching system. Then, requested objects are returned from the cache if those requested objects are found in the local cache. Otherwise, client requests are sent to the original web servers on behalf of the clients. When the objects are requested the first time, the web caching system has to send those requests to the original server. Client waiting time for an object or object retrieval time for a client takes a long time especially when web resource utilization is high (peak web usage times). Thus, web performance becomes poor during peak web usage times.

All client requests are sent to the proxy server. If the requested object is found in the local cache (object hit), it is returned to the client. Otherwise (object miss), the client request is sent to the original web server on behalf of the client. Client waiting time takes longer in the case of object miss compared to that of object hit.

In addition, some web requests can be predicted when the clients are working under the guidance of a leader. Their web requests will be highly overlapped during a particular time interval. Employing the web cache improves web performance for the client requests that are returned from the local proxy cache. When many client requests are returned from the local cache, web performance will be much better. But, the first object request has to be retrieved from the original web server. When the first object request occurs during the limited and narrow network bandwidth times, the user perceived time of the object goes higher. Bringing the possible objects in advance to the local storage will improve web performance (less latency, and object availability).

Since web requests are made at any time (web resources usage at any time), network traffic usually is not evenly distributed. While most people are not at work (or not using the web), web resources are usually lightly loaded or even idle. Web resources are more balanced when light network traffic times are used to bring those (high possibility to be requested in the near future) objects in advance than that of direct web requests for client on demand. Eventually, making a web request for a certain time interval will lead to using web sources more effectively and achieving better web performance. In the following section, web performance will be discussed briefly.

## 2.2 Web Performance

Web performance, which is defined from the client's point of view, is a measure of client waiting time to retrieve the requested objects. Web performance from the client's point of view consists of both web server and object transfer effects. However, most of the researches to improve web performance have been focused on either the web server side or the object transfer path (communication medium, communication protocol, network topology, etc.). Thus, web performance improvements are heading in this direction. In this work, web performance means web performance from the client's point of view unless it is specified. Details of web performance will be explained later.

When web caching is employed to improve web performance, new performance metrics are introduced to evaluate cache and web performance. One of them is the ratio of retrieved objects from the cache to total client requests. This (hit, or object hit) ratio

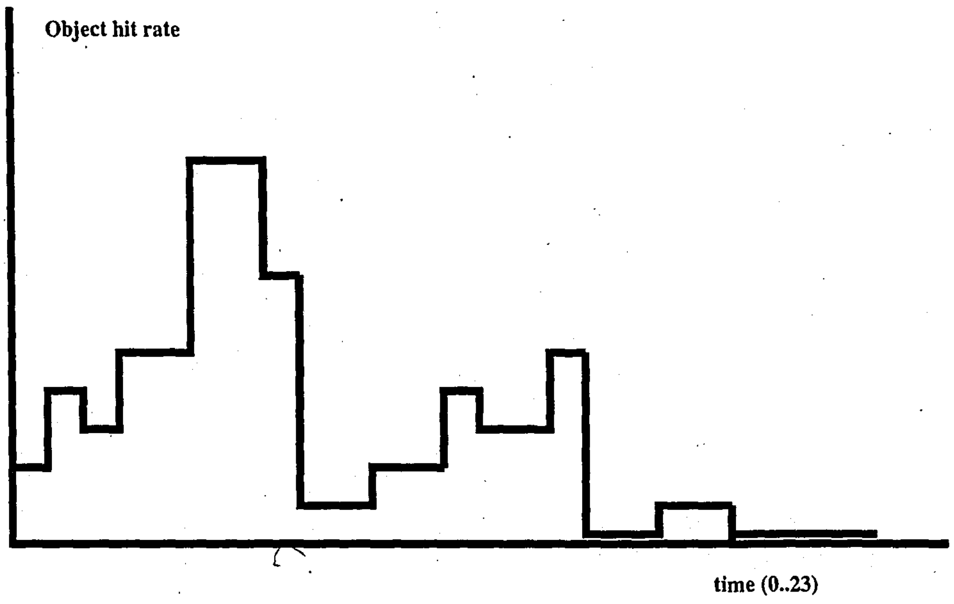

Figure 2.2: Sample object hit ratio throughout the time interval

shows how many times network resources are used between clients and the cache server, and the cache server and web servers. Having a high hit ratio may not imply better web performance.

During the daytime (e.g., 9AM - 6PM), a large number of small object size requests are retrieved with a high hit ratio. During the night (e.g., 6PM - 9AM), a small number of large object requests are retrieved with the same hit ratio. Since each time interval requires different web resources, object hit ratio is not enough to express web/cache performance.

Having high object hit ratios for small and less latency required objects might not reflect better performance. Web performance is considered as hit and byte hit ratio for many other researchers when the web caching is employed. Thus, some studies and implementations have considered average latency as performance metrics. Size of the object and object travel path to reach the client side are major factors in determining how much the client needs to wait. Definition of latency and derivation for some cases are discussed in detail later.

Latency has more meaning as a measurement of web performance from the client's point of view than other performance metrics. Latency depends upon available communication bandwidth between clients and web servers, communication protocol, communication medium, web server speed and capacity, and requested object size.

Most of the latency measurements are based on the average, and medium. Since web resource consumption is not balanced throughout a unit time (e.g., day, week, etc.), latency for a particular client request may be different depending upon when it is made (e.g., day or night, weekday or weekend, etc.). Web caching gives an equal opportunity to each client request to serve either an object hit or miss. Although object retrieval time and object keeping cost in the local cache may vary for each object (large objects require more time and local storage space than smaller ones to retrieve), new performance measures should be searched. Finally, object retrieval time from the client's point of view is chosen as an important performance measure in this work. It considers all factors to deliver a web object to the client from the web server or other places.

In the following section, our motivation for this work will be explained in detail.

## 2.3 Motivation of This Work

The purpose of this work is improving web performance from the client's point of view. The purpose of the web is to exchange digital information between the clients and the servers (direct or via a proxy server) through the Internet. Web servers/proxies are sources of digital information and the Internet permits data transfer from one point to another. Having high client satisfaction shows how effective the web is to exchange data. Thus, reducing latency will lead to higher client satisfaction and better web resource consumption implicitly.

When an available client network bandwidth to the Internet becomes narrow, waiting time of clients gets higher and becomes more important. In addition, web server speed

Figure 2.3: Object retrieval diagram

and capacity also effects client latency. The solution to the problem is having fast web servers and broad communication paths. As a consumer, having fast web servers are out of the client's hand. Improving communication techniques and replacing broad network bandwidth may take a long time and require supplies. Even though having broad and fast network bandwidth may not improve web performance in case of overloaded and slow web servers. Even having enough web resources may not reduce client waiting in the case of lack of resource coordination.

The other well-known solution is web caching which is defined as keeping frequently requested recent web documents (objects) in a place close to the clients in case of serving those objects in the near future. Web caching costs much less time and material to employ compared to the previous solution. Web caching is usually practical and affordable. The latest version of the object stays in the cache unless the requested web object is not reachable or not valid for any reason. Detailed information on web caching will be discussed in the following chapter. Web caching requires a server to process web requests and keep track of objects, and a local storage (disk) to keep the objects. This requirement will cost much less than upgrading communication lines and servers.

Since all the web requests have to go through the cache server, the first access time of the objects is a little longer than requesting directly from the web server because of the cache server process time to determine whether the requested object is in the cache or not. Thus, web performance depends upon when the first access request is made. If the request is made during the peak web working times, web caching may not be enough to reach higher web performance especially if most objects are retrieved from the original server. Thus, the client who makes the first request of the object still has to wait long whether web caching is employed or not. The cache server may return stale (old) objects to the clients if the requested objects are not updated regularly. When the local cache capacity is limited and small, object replacement policy has tremendous impact on cache and web performance. Those web caching issues are discussed in the following chapter in detail.

In addition to the web caching, predicting web requests has the opportunity of reducing the first access time. Detailed information will be given in the following chapter. In short, accuracy of prediction and how to bring those predicted objects to the client/proxy are introduced and have significant impact on web performance. Almost all of the previous prediction techniques bring those predicted objects while a client request is processing or waiting response from the cache or web servers. The problem is how to reduce latency. If prefetching is done while the current client request is processing, prefetching consumes web resources as well. Even if the accuracy of object prediction goes higher (perfect prediction), the client waiting time may be longer than that of a direct connection to the server since some portion of web resources are consumed by prefetching. In conclusion, prefetching introduces additional delays to latency and consumes limited web resources negatively in most research results. If there is a perfect prefetch, its advantages are still limited. Most prediction techniques use either client access patterns or object popularity. When those change suddenly, their benefits may not be significant. Those techniques also consider all clients' requests and objects to predict the future requests by considering long time duration to decide predicted objects. Some of requests or objects may not give the right direction to predict future objects especially those changing rapidly. Some of them even predict objects by using embedded (hyper linked) objects in the current web request [MC98, wco]. The efficiency of this technique depends upon how well clients follow the embedded objects when it is employed.

**Why This Work Was Necessary?**

The following issues for content distribution via the web to achieve higher satisfaction especially from the user's point of view have not been raised in research and practical efforts (works).

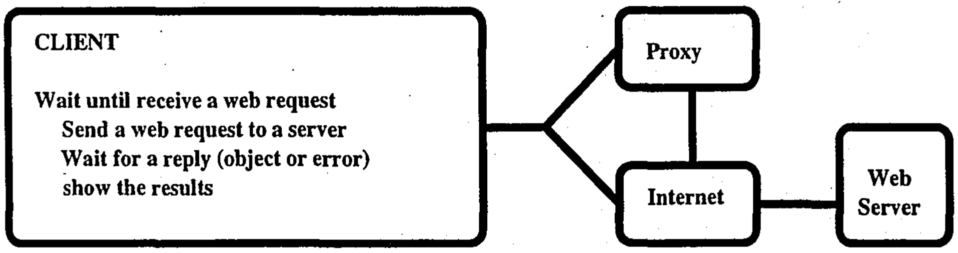
CLIENT
Wait until receive a web request
Send a web request to a server
Wait for a reply (object or error)
show the results
Proxy
Internet
Web Server

Figure 2.4: On demand client request block diagram

1.Web requests of a group of clients working under the guidance of a leader are highly overlapped for a time interval.

2.Prefetching web objects without using previous client or object request information.

3.Retrieving prefetched objects by sending predicted web object requests while current client requests are not processing has not been discussed and studied.

4.Some web object requests such as scheduled web activities can be done in advance.

5.Sending web requests while utilizing web resources (both network and web server) has the opportunity to consume web resources more effectively.

6.Some objects may not even exist when client web requests are placed.

7.Web objects are requested for a time interval to use.

8.The latency as a web performance from the user's point of view is defined because it consists of web server and network performance factors.

As a conclusion, previous studies and implementations have not determined, approached, resolved, or even touched any one these issues. Therefore, this work fills a void by achieving higher web performance and client satisfaction especially from the user's point of view and implicitly from all points of view.

## 2.4 Our Approach

**Direct versus Advance Web Requests:**

Figure 2.3 shows following steps to complete a client web request. It follows steps in order A, C, D, E, F, B.

Direct web requests are made and completed by the client on demand at any time. Web servers are always in the waiting state to accept web requests. Web requests and responses are transferred between clients and servers as soon as possible. Even most of the prefetching techniques make their requests while clients are already making their web requests. When web resources are limited, waiting time of the client will be higher by prefetching because of consuming web resources while client is already making a request.

Web requests can be predicted in some cases. For example, a class has a web activity on the computer lab during a time interval. Most of K-12 schools have limited network bandwidth access to the Internet. Then, waiting time of the students is a big bottleneck during the lab hours (usually during the school times). Employing a cache server at the school will reduce network traffic for repetition of the same web requests. However, the first object access delay still remains. Since schools have some planned (deterministic) web requests for some activities on the web, some of the web requests can be estimated with high probability. The instructor knows that the class will visit possible web sites and web objects. Most of the prefetching methods use either client request patterns or object popularity to predict near future client demands. When client requests do not have any relation to the previous web requests, their prediction accuracy will be insignificant. In addition, most prefetching techniques send predicted web object requests while client requests are processing. Thus, conventional prefetching techniques do not only increase latency for all web requests but add extra load to the entire web request path as well.

Figure 2.4 shows the steps to complete an advance web request. Multiple client requests may be combined. The following steps to complete prefetching need to be taken in order whenever web resources are available to consume effectively.

When a group of clients work under the guidance of a leader and have scheduled web activities, their web requests can be done in advance. The leader can predict objects

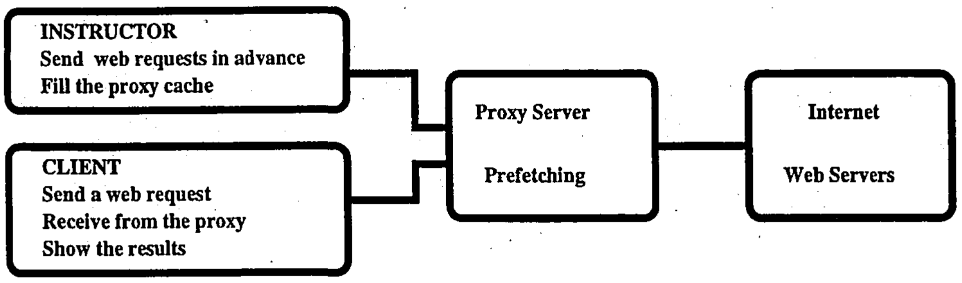

Figure 2.5: Advance web request block diagram

(with high probability) before each group member makes a request. This web request is different from conventional client requests. The instructor initiated prefetching is introduced in detail as a special case of advance web requests in Chapter 4. In general, some web requests can be done before actual client requests are made. Since prefetched object requests are completed any time before any one of the clients makes any request, prefetching is done by using web resources effectively. Those prefetched objects have to be valid for a time interval. Although it is necessary to scheduling object requests at the proxy server, the advance web request method has many advantages over the direct connection approach. Some of the benefits are given briefly below. This work is challenged to search and represent how advance web requests via prefetching and proxy cache are beneficial to improving web performance from the user's point of view by using web resources effectively.

**Kernel of Advance Web Requests**

The advance web requests while using web resources more effectively via prefetching have three parts:

1.Predicting web requests (objects) for a group of users working under the guidance of a leader. The leader decides predicted objects. The intelligent web request handling mechanism for predicting objects is vital for this approach since effectiveness of the prefetching depends upon how prefetched documents are found.

2.Retrieving those predicted objects while web resources are used more effectively is another factor. Collect objects from the web servers while the web servers and the Internet are lightly loaded. Thus, web resource consumption is different from the previous studies. It does not bring any additional overhead while the current clients requests are processing.

3.Prefetched web objects should be available during a time interval.

Influence of this method for a group of users when they work under the guidance of the instructor reflects how well users follow the instructor and how effective the proposed approach.

Web performance is considered from the user's point of view as latency (client waiting time, or object retrieval time). It is explored in detail in Chapter 5.

**Distinctions From Other Studies**

Distinctions of this work from other studies are grouped for:

1. The Instructor Initiated Web Caching:
    (a) Better prediction technique.
    (b) Prefetching predicted objects while web resources are lightly used.
    (c) Prefetched objects need to be valid for a time interval.
    (d) Web performance (client waiting, object availability, client satisfaction) is better.
    (e) Web resource utilization especially for prefetching is much better than others.
    (f) Advance web requesting for a time interval is a unique and novel approach.
2. Web Performance:
    (a) Client waiting time, object retrieval time, or latency are considered a web performance measure. Web performance from the user's point of view consists of web server and the network issues implicitly.

(b) The instructor has an important role in achieving higher web performance since the instructor decides the predicted objects.

(c) Web resources are used effectively to retrieve predicted objects.

(d) Prefetching reduces web requests between proxy and web servers.

(e) Prefetched objects are kept in the local cache for the time interval.

(f) Client requests are returned from the cache with a higher object hit probability. Because most of the web requests are returned from the proxy cache, object retrieval time be will reduced for that group of users for a time interval in case object hit happens. This approach reduces web request traffic implicitly in the overall web system.

(g) When latency is reduced, client satisfaction will be better than other approaches.

3. Interest Groups:

(a) A group of users work under the guidance of a leader have common web requests during the time interval.

(b) The leader has a great influence on a group of clients.

(c) Prefetching for a group of users for a time interval is a novel approach.

(d) Web resources are used effectively (prefetching, keeping and finding objects, and overall implicitly).

(e) Web performance is devoted to more issues (client, proxy, network, web server) implicitly.

(f) Generalization of this approach for the same interest groups within the same proxy has the potential to achieve better web performance.

(g) Generalization of interest groups in the web caching hierarchy has the advantage of sharing common objects for the same interest groups.

(h) Many interest groups in the same proxy or in the cache hierarchy are generalizations of this approach.

(i) Analysis of the latest novelties is not trivial. New tools need to be developed.

4. Advance Web Requests:

   (a) Web requests can be done before they are actually needed although current web requests are client on demand type. The client makes a web request and waits to see the requested objects right away.

   (b) Objects have to be valid for a time interval for advance web requests unlike current implementations that require the object validity when the request is made.

   (c) Objects may not be valid when the advance requests are placed. The web request is still valid. On the other hand, current implementations require that the requested object has to be valid when the request is made.

   (d) Web resource utilization can be used effectively and efficiently.

   (e) Advance web requests change object requests and distribution on the web. Further, it will affect information exchanged in general.

   (f) Web performance as latency, client satisfaction or resource usage should be considered to make comparison with others.

   (g) The difficult part of this work is evaluating this novel approach and comparing with other methods.

The most exciting part of our motivation is advance web requests. The philosophy of web requests should be changed from client web requests at any time to advance web requests within a time interval when clients are working under the leader. The conventional web request is defined as retrieve the object as soon as possible. However, this work is devoted as retrieve the objects by the beginning of the time interval that clients start requesting and keep valid objects until the required time interval is resumed.

Although clients know what they want in the near future, web objects or resources may not be available when they make their requests. For example, the client reads a particular newspaper or magazine periodically (e.g., daily, weekly, monthly). Making a web request for the daily newspaper will bring the web objects before either the client makes the request or the objects are not available yet. This request can be done during the night after objects are available. Making scheduled web activities for a group or a client will change the nature of web requests. Making advance web requests will lead to using web resources more effectively. Eventually, the waiting time of clients will be reduced.

Sometimes clients are interested in some objects as soon as they are available. For example, the latest Red-Hat Linux distribution needs to be retrieved as soon as it is available. It is too big and its server may be overloaded when the client makes the request. In addition, the client may not know when it is going to be available. Making a web request for the future objects is not possible with the conventional approach. However, making an advance web request will reduce client waiting time especially if large web objects are demanded by many other clients. Having a mechanism to make an advance web request and distribute it to an interest group will help to reduce latency.

## 2.5 Novelties From This Work

This work explores novel web object requests, web resource utilization, web performance, object dissemination, via the Internet by multiple clients, objects, proxy servers, and web servers.

Hopefully, conclusions of this work contribute most of the distinctions given above explicitly, or even implicitly in part.

## 2.6 Judging This Work

Conventional web performance measures do not give enough information, but this work explores the following:

Since the nature of conventional and advance web requests are different, comparison of those methods is not appropriate by using performance measures that are developed for the conventional techniques.

1.Client latency is defined as a retrieval time of on demand client requests. Latency consists of retrieving objects from the server to the cache and retrieving from the cache to the clients. In advance web requests, retrieving time of the object from the server to the cache cannot affect waiting time of clients. The only client waiting time is retrieval time from the cache to the clients.

On demand: The client sends the request and waits to get reply. Retrieval time of the object is either from the server or the cache. All times are counted while the client request is processing.

Advance demand: Some object requests are made in advance. Then, the client sends the requests and most probably gets the object from the local cache. Client waiting time is retrieval time of the object from the cache.

2.Prefetching accuracy:

- Accuracy is determined by how well clients follow the leader (e.g., prefetch hit factor is the ratio of prefetched hits to total hits) during the time interval. Conventional methods usually measure hit ratio for long periods (e.g., considering all requests).

- Another measurement would be the prefetch size factor, which is the total of prefetched byte hits to the total byte requests from the cache for a time interval.

- Analyzing prefetched hits / bytes and prefetched objects is another way to see effectiveness and efficiency.

- Object retrieval time for prefetched objects and object popularity is another way to see the usefulness of this method.

Most of the cache performance studies have been done for all clients and objects. Our approach is focused on a group of clients in a particular time interval; all of the measurements can be done for a group of clients with a time interval. Our approach is to prefetch some objects for a group of clients, but not all of the clients. Therefore, some clients may not benefit explicitly and this method may not be useful all the time. Efficiency of the method should be considered for a group of clients for a time interval.

3.Client satisfaction because:

- Clients have fast object retrieval time from the cache.

- Clients have the opportunity to make web requests in advance even if the objects are not available.

- The novel cache replacement policy for the prefetched objects during the time interval will speed up the cache processing time and lead to better resource usage of the local storage.

- Sharing the same objects with other interest groups will achieve better resource utilization and reduced client waiting time.

- Making planned web activities will to optimize the web resources more effectively.

- Prefetching while web resources are lightly used will achieve effective web resource consumption.

- This approach will reduce the cost of web resources.

- This approach will support content distribution to interest groups in advance.

## 2.7 Conclusion

The innovation of this work is making advance web requests to utilize web resources more effectively and efficiently to achieve higher client satisfaction. Making web requests for a group of clients, and delivering those web requests while utilizing resources more effectively are important aspects of this work. As a result, the content delivery mechanism to the interest groups with time interval will be new avenue to reach higher web performance and content dissemination.

# Chapter 3

# Related Work

*Stairs are climbed step by step.*

In this chapter, previous work on the web caching is discussed in detail. Then, prefetching web caching techniques are introduced briefly. This part of the work explores the fundamentals of the web caching and shows the necessity and importance of our work. Some of the open research problems and the basis of our approach are also given.

## 3.1 Overview of Web Caching

The caching concept was first introduced in computer architecture systems, as memory caching. Memory caching is defined as the most frequently used data are stored temporarily close to the processor to be able to minimize data retrieval time. Because data retrieval time from cache memory is much faster than that of conventional memory, caching gives an opportunity for higher performance. By a similar analogy, web caching is defined as bringing a copy of frequently requested web documents close to the clients in case they are needed in the near future, so as to minimize document retrieval time and maximize throughput.

Web caching differs from memory caching in many ways such as data size, data accessibility, speed of data access, and cost of data. Cached data size varies from a few kilobytes to a few hundred megabytes in web caching, whereas data size is almost constant in memory caching within a few bytes. In addition, web objects may not be available at all times because of overloaded servers or server failures. However, data are always reachable in memory caching. Speed of data access in memory caching is much faster than that of web caching: data access speed in memory caching is nanoseconds (depends upon cache memory speed); but data access speed in web caching is larger than milliseconds or even seconds (depends upon cache server speed, local storage speed, retrieval time between cache server and clients). Even if the principles of memory and web caching are similar; working environment, expectations, and properties are quite different. Some web caching research has been done by using conventional computer caching [LA94].

Web caching has four main advantages. First, web caching reduces the network bandwidth consumption of web server and client – between institutional network and regional Internet Service Provider (ISP), between transoceanic or international network and regional network [LOG96]. Optimization of network bandwidth usage will reduce network costs. Even if the network infrastructure stays the same, upgrading network links will be delayed in the long term. In overall, efficient network bandwidth usage allows an opportunity for cheaper and faster network communication. Second, web caching reduces the web server workload. Efficient usage of web servers not only increases serving capacity but decreases delays and queuing at the web servers as well. Therefore, the web server will use an inexpensive server. As a result, fast and high web server capacity will be available to the clients. Third, web caching reduces latency by avoiding slow or congested links between client and web server. It uses those slow and congested links only once or when it is necessary. In addition, popular (frequently requested) web objects are retrieved from the cache server. In general, waiting time of the clients (latency) will be reduced because of some portion of web requests returned from the cache server. Other requests use a less loaded communication network and web server resources. Fourth, web caching gives object reliability; when the web server is not reachable, all of the valid objects or the latest version of the objects may be retrieved from the cache server [Flo98].

Although web caching has great benefits, it also has problems. First, cache contents need to be updated when contents of objects are changed at the web server. Cache consistency that have updated (fresh) objects not only increases the hit rate (e.g., implicitly cache performance) but also avoids returning stale objects [Cor96]. However, keeping objects consistent in the web cache requires additional network bandwidth consumption and bookkeeping overhead. Second, some web objects are not cache-able such as dynamic web objects (generated for each web request (e.g., Common Gateway Interface (CGI) [McC94] script returns). Since conventional web caching methods do not support dynamic object caching, some research should go this direction. Third, when a web object requires user authentication, that object is also not cache-able. This is called point to point private data exchange through the web. Our research is focusing on distributing publicly and freely available web objects on the web. Fourth, web servers may not want to give a chance to cache objects because of keeping track of object popularity (page hits). Fifth, when an object cannot be found in the cache server, waiting time of the clients via the cache server will be slightly higher than that of direct connection to the web server. The cache server has to check whether that object in the cache or not. If that object is not in the cache, the cache (proxy) server requests that object from the original server on behalf of the client. Thus, the cache server introduces additional latency for missed objects.

What are the performance measurements in web caching?

Performance can be divided into three parts depending upon the point of view. First, from the client's point of view, performance is defined as a waiting time of the client, which is the total amount of time required to bring the object from web server to the client. The client always wants to get the object as quickly as possible. Second, from the web server's point of view, web performance is balanced in terms of the web response or request to the Internet. As long as a web server replies to each one of the web requests immediately without keeping it in a queue or jeopardizing its role, it will perform its function. It is not always easy to balance web requests because client web requests can be made at any time. Third, from the network's point of view, performance is balanced for the network traffic whenever using a minimum of network resources for each individual web transaction makes a web request. Again, web requests can be made at any time, and it is difficult to balance network traffic and network usage [VdJM98, HiCYO99].

When web performance is discussed in the following sections, performance from the client's point of view is considered. The purpose of this work is to reduce latency while utilizing web (network and web server) resources more effectively. Reducing client waiting time will lead to higher network and web server resource utilization and better client satisfaction. Web performance will be discussed in the following chapters in detail.

Cache performance is based upon how effectively and efficiently the cache server is used to satisfy client requests. Object hit ratio, object byte hit ratio, additional introduced delay by the cache server, cache speed which consists of request processing speed, searching the requested object in the local cache, reading object from the local cache, and server capacity and the communication bandwidth between clients and the cache server are some cache performance measures. Those metrics are used in the previous research studies.

In summary, some research has been done to get benefits from web caching. There are some factors which have a significant impact on web cache performance such as maintenance of web cache, design of web caching system, etc. In the following sections, principles of web and communication protocols will be surveyed. Web caching improves web performance since web resources are consumed effectively and help to balance web resource usage in the web.

## 3.2 How does Web Interact?

Web interaction between clients and web servers will be briefly summarized in this section. The web object transfer protocol between web servers and web clients is called Hypertext Transfer Protocol (HTTP) [BLFF96, Fea97]. Web servers are always in wait state. When a client's request comes to the web server, the web server parses incoming message. Then, the server responds according to a request method with a reply code. The request method could be GET, POST, etc. For example, a client requests an object from a web server. The client's request must include a request method, which is GET, and an object name which is defined by Uniform Resource Locater (URL) [BLea94].

Client: `http://www.eecs.lehigh.edu/index.html`

Client server: sends a message to web server 'www.eecs.lehigh.edu', adding HTTP header (request method is 'GET') and object name 'index.html' through the Internet. As soon as the client server receives the reply message from the web server, it is displayed to the client. The client server is also called a web browser.

Web Server: It awaits a client request. When a client request comes, it parses client request such as request method 'GET', web server host name 'www.eecs.lehigh.edu', and object name 'index.html'. If that object is available to this client, the web server returns the object with the reply code and waits for another request.

## 3.3 Summary of HTTP

HTTP is an application layer protocol that exchanges messages between clients and web servers via TCP. HTTP protocol is composed of two parts, which are HTTP request and reply. HTTP request consists of a request method, a uniform resource locator (URL), and a request header. HTTP reply consists of a numerical result code, a set of reply headers, and an optional reply body. There are two versions of HTTP, which are HTTP/1.0 and HTTP/1.1 [BLFF96, Fea97]. Some recent web applications are using HTTP/1.1 because of its advantages and better performance [KW00].

HTTP is simple, extendible and fast data transfer protocol over TCP. HTTP server is stateless and messages are carried in ASCII form.

TCP is a reliable byte stream, connection oriented, and stateless data transfer protocol. There are two versions of TCP, which are TCPv4 and the latest one, TCPv6. More information about TCP can be found in Stevens' book [Ste98].

CERN-httpd and Apache are some of the well-known web servers.

Lynx, Mosaic, Microsoft Explorer, and Netscape are some of the well-known client servers (browsers).

## 3.4 Web Caching Architecture

Web caching system that is also called proxy server or cache server is located between clients and servers, usually close to the clients. When a client makes a web request through the cache server, the web object is either in the cache or not. If the requested web object is found in the cache (object hit) then the object is returned to the client from the local cache. Object retrieval time from the cache is much smaller than retrieving documents from the original server (direct request). On the other hand, if the object is not found in the cache (object miss) the cache server requests the object from other cache servers or directly from the original web server on behalf of the client. As soon as the

Figure 3.1: Object retrieval diagram

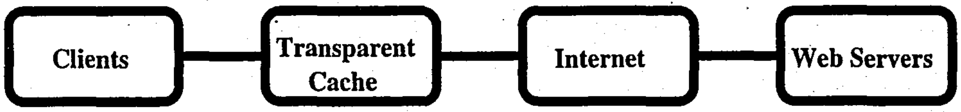

Figure 3.2: Transparent cache

cache server receives the object, it sends a copy to the client and keeps a copy in its local storage. When an object hit occurs, waiting time of the clients is much shorter than that of direct connection to the original server because of retrieving objects from the local cache. But, object retrieval time is slightly longer than that of direct request from the original web server when object miss occurs because of spending time to look for the object in the local cache.

A web object is referred as a document available on the web such as text, image, audio, HTML, postscript files, etc. via HTTP in the Internet. The Internet and network are both defined as a connection among computers (clients and servers) all over the world unless it is specified.

There are two types of single web cache architecture: transparent and proxy cache.

A transparent web cache acts like a firewall or gateway. It is located between the client and the network. It is useful when the network cannot flap between routers. Clients cannot easily bypass the transparent cache except by network routing flaps so that no data flows through the cache [Wil98, CAJ+99]. The client has to go through the transparent cache server whether the client likes it or not. Since transparent cache is not flexible and

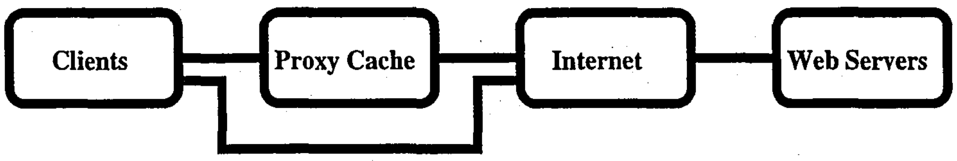

Figure 3.3: Proxy Cache

scalable, there is no research or work done for vital applications. The transparent cache server's function is explained above 3.2.

A proxy web cache is located between the clients and the web servers. When the client requests an object, the request goes to the cache server. If the proxy cache holds the object, it returns the web object to the client. If it does not, it requests the object from the web server on behalf of the client. The web server sends the object to the proxy cache server. Then, the proxy cache server sends the object to the client and makes a local copy at the same time (called store-and-forward). The major difference between the transparent web cache and the proxy web cache is that the client can easily bypass the proxy cache any time, e.g. [LA94]. Most of the commercial products can be found at `http://www.web-cache.com/products.html`. A survey on some of the well-known proxy cache servers can also be found at [Cor96].

Web cache, proxy cache, and cache are used for the same meaning unless it is specified in this work. In the following section, cache miss will be discussed briefly.

## 3.5 Cache Misses

If an object can not be found in the cache, it is called cache miss (object miss or miss). There are five reasons for cache misses – compulsory, capacity, communication, uncachable, and error misses [TDVK98, RRB98].

The compulsory miss happens when a web object is requested for the first time. Push caching and prefetching may decrease compulsory misses. They will be discussed in this chapter in detail. The capacity miss occurs when an object has been discarded from the cache to make space for other objects. Increasing cache storage capacity and optimal cache replacement techniques can reduce the capacity miss. The communication miss occurs when an object has been invalidated due to an updated object. Its effect can be eliminated by push caching when the web server updates objects in the cache by sending a recent version of the objects. Push caching techniques will be discussed later in this chapter. The uncachable miss occurs when the cache server must contact the web server to retrieve the object. When the object is generated dynamically, it is hard to expect to get the same client request return each time unless the cache server knows client input and possible replies from the web server under the relatively stable (special) conditions. However, previous researches have been focused on static web objects which do not change for a long time and are cache-able. No research has been done on this topic. There are open research opportunities on caching dynamic objects. Advanced web requests solve this problem by keeping the prefetched object for a time interval. In addition, there more work needs to be done. The error miss comes from error reply of the client request. Finding the reasons which caused the error and eliminating those reasons can eliminate error miss.

## 3.6 Cache Replacement Policy

Because cache storage capacity is limited, the cache replacement management has to move one or more objects to make room for recently requested objects when the cache storage fills up. The cache replacement policy is one of the dominant factors for higher cache performance. Some well-known cache replacement policies are as follows:

LRU (Least Recently Used): Least recently accessed object will be removed to make space for new coming objects. When client request pattern varies frequently, LRU does not give a good cache performance. It is good for popular objects that are already in the cache. **Squid** proxy server version 2.2 [squ] uses LRU as a cache replacement policy.

LRU-MIN: It is a variant of LRU that tries to minimize the number of objects replaced. In addition to LRU, it gives a good cache performance for large web objects that are already in the cache.

LRU-THOLD: It is a variant of LRU that avoids removing a large number of small objects. In addition to LRU, it gives a good cache performance for small web objects that are already in the cache.

FIFO (First In First Out): First accessed object will be removed first when it is necessary. It is easy to implement and gives a better cache performance for changing client request patterns. If a client request changes often, it may not give good cache performance.

LFU (Least Frequently Used): The least popular object will be removed to make space for new coming objects. Implementation of LFU is not easy compared to above policies. However, it gives a good cache performance for most popular web objects.

LFU-Aging: Access frequency of the object and object age are both considered when removing objects. In addition to LFU, this method gives priority to recently requested web objects. Both object popularity and object access time are counted for cache performance.

GDS (Greedy Dual Size): Object size and cost function play a role in removing objects. When the size of the web object and communication bottleneck is really important, this method gives better cache performance.

GDS-Hints: GDS policy that optimizes popular and small objects.

Many research results compare the variety of web cache replacement policies [KKO98, DAP99, ARA+95, WAAF96].

Each cache replacement algorithm has some advantages and bottlenecks. Finding an optimal web cache replacement algorithm depends on object access pattern, overhead of replacement policy, availability of network bandwidth between the content providers and the cache servers, and between the cache server and the clients. Each one of the cache replacement algorithms gives higher performance than others under the specific circumstances. Sometimes, it is also difficult to find the best cache removal algorithm to employ. In that case, using a combination of more than one cache removal algorithm or switching the algorithm among a group of choices may be the best solution. As a result, flexible cache replacement policy will be a good candidate achieving better cache and web performance.

The implementation of a single web cache is easy and straightforward. It is useful for a small number of clients, which have more common request patterns. However, when there is no common interest among the clients that share the same single cache or the total number of clients is too large to use single cache, using single cache may not be an efficient solution. In that case, multiple web caching architectures may be suitable to get higher cache and web performance. In the following section, multiple web caching methods will be surveyed.

## 3.7 Multiple Web Caches

The multiple web cache reduces network bandwidth consumption, decreases web server load, and improves object retrieval time. Sharing objects in the web caching system plays important roles on the overall web performance; however, Inter-cache communication is the dominant factor in achieving better performance.

There are four design principles for large scale multiple caching systems. First, the caching system should take minimum number of hops to locate and access the web object. This will reduce object retrieval time. Second, the caching system should not slow down on object misses. Since there are many web cache servers on the system, finding available objects on the caching system may takes longer than retrieving objects from a particular cache server or even the original web server. The caching system should handle object misses efficiently. Third, the caching system should share web objects as much as possible. This will be discussed in the following section in detail. Fourth, the caching system should store web objects close to the client. If a distinct client requests a web object, keeping the object close to that client will resolve object storage problems. But, more than one client may request the same web object [TDVK98]. The multiple web caching system should balance finding web objects in the hierarchy and retrieve the objects with optimal communication cost.

There are two multiple caching systems discussed briefly in the following sections. Those are hierarchical and distributed web caching.

### 3.7.1 Hierarchical Web Caching

The hierarchical cache architecture is auspicious for scalability and manage-ability for web caching. A group of child caches share a common parent cache. When two client caches have the same parent cache, they are called a sibling cache. A child cache forwards client requests to the parent cache when the child cache does not have the requested object. Then, if the parent cache has the object, it will return it to the child cache. Otherwise, the parent cache requests the object from higher caches or the original web server on behalf of the client. When the parent cache retrieves the object, it keeps a copy of the object in its local storage and forwards the object to its child. The requested object is forwarded from the top level to the bottom level through the cache hierarchy until it reaches the client who made the request. The difference between sibling cache and parent cache occurs on miss objects – the sibling cache cannot forward the child's request but the parent cache can. Both parent and sibling caches can reply with the requested object if they have it in their cache.

There are more than 2 levels for complex hierarchical web cache architectures. In general, hierarchical web cache topology can be defined by l-level q-ary tree.

Hierarchical web caching achieves more cache hits by using neighboring caches. Routing requests to a specific cache decreases HTTP (web) traffic for a particular path.

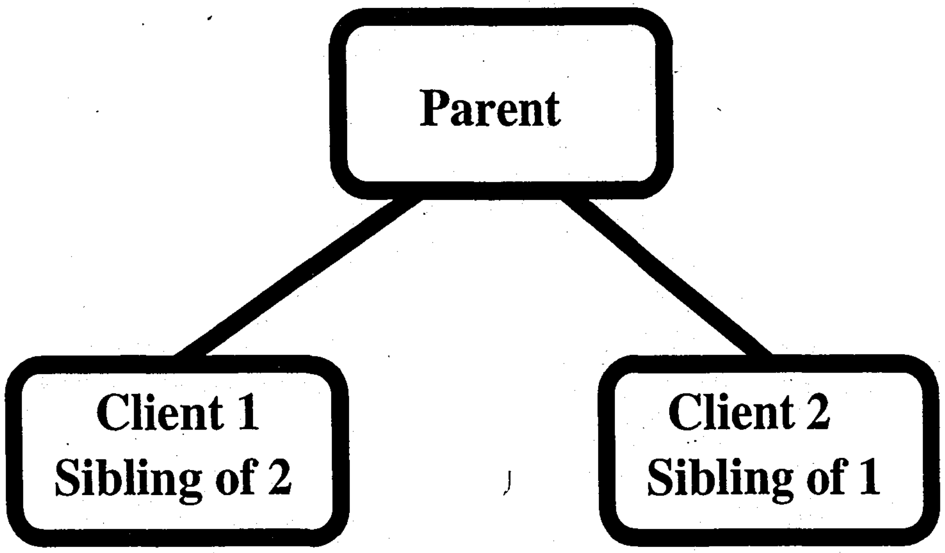

Figure 3.4: Parent, sibling, and child cache

Since there is one copy of each object in each level, caches at the higher levels need a lot of storage space. Because the hierarchical cache system uses store-and-forward method, it requires more storage capacity and increases object retrieval time when the cache system becomes bigger. The implementation overhead is the bottleneck. It needs a TCP connection each time when the web object needs to be retrieved. HTTP/1.1 allows persistent connections between clients and servers. A better solution might be using Internet Cache Protocol (ICP) [WC97] to query neighboring caches. Inter-cache communication is only possible hierarchically through TCP connections. But, caches at the same level cannot be accessed directly. Hierarchy should not have more than 2 or 3 levels according to some previous research results [PrSB99, VdJM98].

The communication among caches in the hierarchy is the bottleneck of the total performance. Hierarchical web caching introduces additional delays by cache misses increasing object retrieval time because of querying the whole hierarchy from the bottom level to the top level. Configuration has to be done in each individual cache as well. The communication protocol on the hierarchical web caching will be surveyed briefly in the following

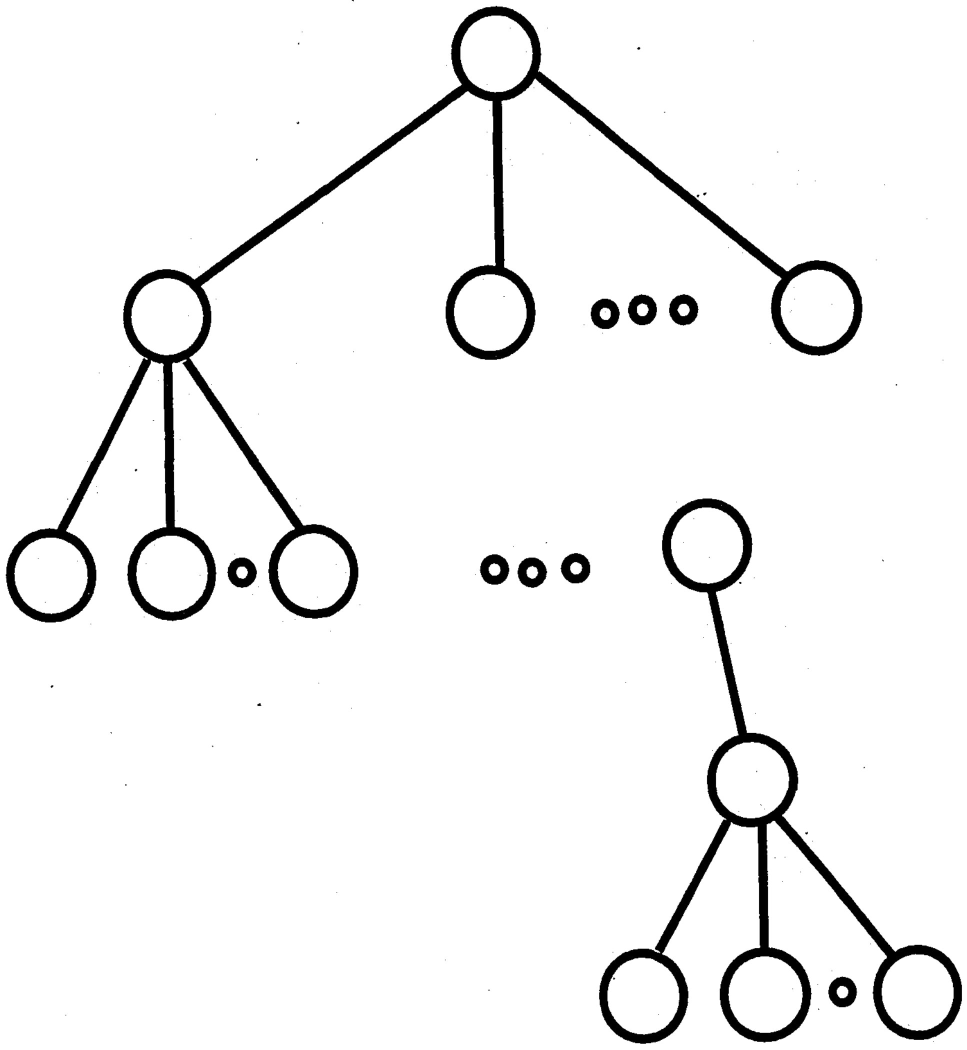

Figure 3.5: General Cache Hierarchy

section.

### 3.7.2 Internet Cache Protocol (ICP)

Internet Cache Protocol (ICP) [WC97, WC98] provides a communication scheme between caches in the cache hierarchy and offers to establish complex cache hierarchy.

ICP consists of a fixed 20-octet header and a variable-sized payload. ICP header format is shown below in the order of header.

OPCODE (8 bits) indicates the messages type such as ICP_QUERY, ICP_MISS, ICP_HIT, etc.

VERSION (8 bits) indicates the protocol version for backward compatibility.

PACKET LENGTH (16) represents total size of the message.

REQUEST NUMBER (32 bits) identifies the client request.

OPTIONS (32 bits) shows optional features and recent additions to the protocol.

SENDER HOST ADDRESS (32 bits) is intended to hold a host IPv4 address of the client.

A cache sends a query message (ICP_QUERY) to its peers. The payload of the query is a size of URL. As soon as each peer receives the query message, each cache returns to the sender with ICP_HIT or ICP_MISS depending upon existence of the object locally. The cache sends ICP_QUERY, collects reply messages and then, selects a peer cache according to a selection algorithm.

ICP uses either Transmission Control Protocol (TCP) [rfc81b] or ewariUser Datagram Protocol (UDP) [Pos80]. ICP currently uses UDP for simplicity of implementation and minimizing object retrieval time. As a result, current implementation of ICP is unreliable because UDP is an unreliable communication protocol.

### 3.7.3 ICP vs. HTTP

Since ICP is simple and light, the implementation of ICP can quickly parse and interpret. ICP turnaround time is faster than HTTP because of unreliable protocol. When message drop out becomes a bottleneck on the data communication environment, ICP performance becomes very poor. In addition, ICP does not match HTTP. Although ICP is used to locate web objects, proxies use HTTP to retrieve the objects.

### 3.7.4 Additional Delays with Hierarchical Web Caching

Though hierarchical web caching has great advantages, it introduces four types of delays. First, resolution delay that is the time to check whether the web object is present in an ISP (ICP query), hashing function, and routing. In order to keep this delay low, the caching hierarchy should not have more than three levels. Second, TCP delay which comes from the start phase of different TCP connections between every cache level. Instead of opening consecutive HTTP requests to the web server, combining a bunch of HTTP requests may reduce TCP connection delays. In contradiction, combining those HTTP requests may increase TCP delay when communication medium drops messages frequently. Third, server delay, which is due to busy web servers that have to serve many requests from several national caches and can be eliminated by broad server capacity. Fourth, queuing delay which comes from queues on busy caches [RRB98]. As a conclusion, there is a trade off between hierarchical web caching and additional delays. Web performance really depends upon combination of advantages and disadvantages of hierarchical caching system.

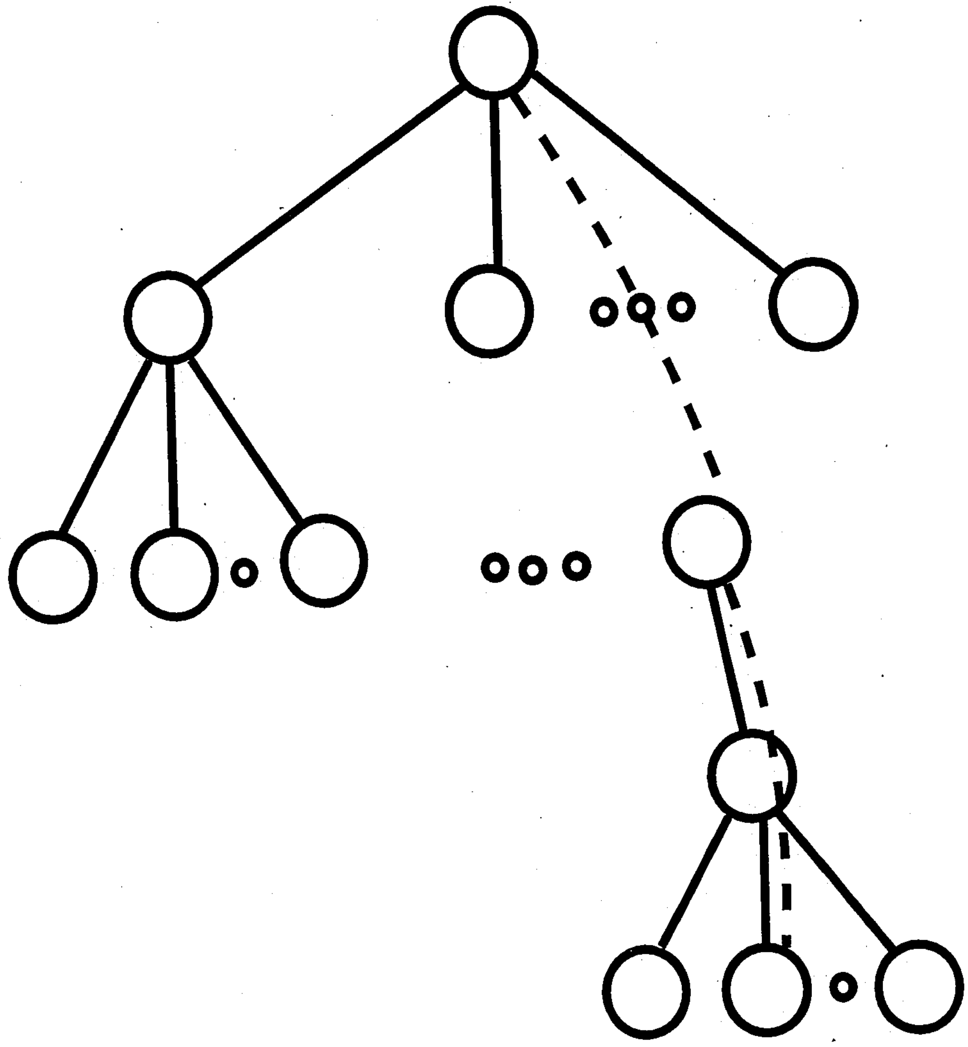

Figure 3.6: A path from the bottom level to the top level to seek the object in the cache hierarchy

### 3.7.5 Distributed Web Caching

Since a hierarchical web caching architecture introduces extra delays, a distributed web caching architecture has become useful to avoid problems with extra delays. Instead of having a hierarchy between caches, there is no restricted network topology in the distributed web caching. The network connection topology of distributed caching is a connected graph. The hierarchical web caching topology can be embedded into the distributed web caching as well. When network topology becomes too complex, the more flexible communication topology becomes practical. In addition, distributed web caching is called cooperative web caching [RRB98]. The communication between web caches is done by Cache Array Routing Protocol (CARP) [VR98]. In the following section, the comparison between hierarchical and distributed web caching is discussed briefly.

### 3.7.6 Hierarchical vs. Distributed Web Caching

Hierarchical web caching has shorter connection time than that of distributed web caching because of keeping additional copies in each level unless distributed web caching has fully connected network topology (because of implementation cost, it is not practical). Because of additional copies at the intermediate levels for hierarchical caching, hierarchical web caching consumes more local cache storage capacity than that of distributed. Distributed web caching may or may not consume higher communication bandwidth than hierarchical, depending on object request patterns and availability of the objects. Since distributed web caching uses more communication bandwidth in lower levels in which they are less congested, the traffic generated by distributed web caching has more distributed network traffic throughout the web caching system. No more than three levels for hierarchical web caching would be suggested [PrSB99, VdJM98].

Hybrid web caching that is a combination of both hierarchical and distributed web caching might be the best solution for general purpose applications to achieve better web performance.

### 3.7.7 Improvements on Conventional Hierarchical and Distributed Web Caching

Although hierarchical and distributed web caching solve some problems, they also introduce new bottlenecks such as finding an algorithm to retrieve a cached object in the cache architecture as quickly as possible, and decreasing the object retrieval time by deciding whether to search through the cache architecture or go directly to the web server. For example, the caching architecture is too complex and communication between cache servers has some topology. When a client requests an object, it goes through its cache server first. If the object is available in there, it returns object hit with the requested object. If not, it asks the cache hierarchy to determine whether any one of the cache servers has the object. Then, the cache server waits until one of the peers reply with cache hit or decides that no one has the valid object after a certain time. Then, the requested object is retrieved from the web server. Deciding whether an object is in the cache or not takes a lot of time and consumes a lot of network traffic and computation time in each cache server. The cache architecture must maintain object consistency as well [ISY98].

T and et al. [TDVK98] looked at the problem that location of the object in the entire cache architecture should be known to retrieve the object directly from the cache. Although, this eliminates object miss on the cache architecture, it requires additional data storage, which is called meta-data file, to locate the objects. Because their work needs extra data storage capacity in cache servers, cache consistency problem is eliminated in the cache architecture by meta-data file. Since web objects are also stored at the leaves of the hierarchy, cache-to-cache object retrieval may take longer than conventional hierarchical approach. Their method not only minimizes the object retrieval time on object misses by using meta-data to locate the object but also decides quickly whether it is an object hit or miss. One of the disadvantages of the meta-data file is that it may be too large when the cache architecture has many levels and caches. Another disadvantage of this method is that consistency of the meta-data file is a bottleneck. When this file becomes too large, maintenance of this file does not only require extra network bandwidth, but also requires extra computation to keep track of the cache status and objects. Although, this method promises an opportunity for better web performance, the meta-data is the important bottleneck. It may not be suitable for the optimal solutions in many circumstances.

## 3.8 Push or Prefetch Web Caching

The web performance not only depends upon network and web server components but also depends upon client access patterns; those are changed by time. Many people did research on web caching to look at one or a few parameters to gain better web performance. Many research studies and implementations have considered current client requests to predict near future client requests [GS95, Gwe95, LOG96, Tou98, PM96, DCW96, IKY97]. Another approach to improve web performance is predicting client requests with an intelligent prediction model or sending objects to the cache system when they are changed at the web server. Those techniques are called prefetch and push caching which estimate future requests and update the objects based on the current and previous client and web server profiles.

One of the approaches to get higher web performance is to give more responsibility to the web server. Whenever the web document changes, the web server informs or updates objects held in the cache architecture by using object or client access patterns. When the web server tries to update the objects in the cache before clients make any requests, it is called push cache. Push caching does not only increase the hit ratio and byte hit ratio but also improves object retrieval time. However, push caching can increase the network traffic and web server load.

There are four push or prefetch caching design criteria. First, the algorithm is efficient if the large amount of copies that are made actually succeed in improving cache system response time overall. Second, the algorithm should provide performance gain. An efficient algorithm pushes a small number of objects with high request probability in the near future.. Third, the algorithm should be adaptive. If the algorithm runs in a resource rich environment, it should aggressively replicate objects to improve response time even at the cost of efficiency. Fourth, the algorithm should be easy to implement and avoid consuming too many resources figuring out what to push [TDVK98].

There are two types of push caching: web server (or server) initiated web caching (push caching) and client initiated web caching (prefetching).

### 3.8.1 Server Initiated (Push) Web Caching

When the content of the web document is changed, clients will be informed by the central service. The web server initiated web caching works according to the access pattern of the object (popularity) at the web server side. It is also called push caching. The web server keeps track of the popularity of each object. When an object is popular enough to be pushed, the web server sends that object to the web caches as soon as the object is updated. Push caching increases network traffic and object hit rate as well. Web performance depends upon the pushing algorithm, object popularity, and how much network bandwidth is consumed. However, deciding whether an object is popular or not is an open research problem. If the local object popularity varies abruptly, the web server push caching is not the best solution. On the other hand, it may be good for long term object request trends.

Server initiated push caching by geographical location of the cache server, request pattern, and object popularity has been studied by [GS95, Gwe95, LOG96, Tou98]. Because the web server has network topology and access history of objects, object requests should be clustered geographically. According to their research results, the access pattern of web objects is not distributed geographically. However, finding a cache server to minimize network bandwidth may not be trivial. Since the geographical location of the web object is not really important (Internet topology is based on IP address and class of the network), their approach may not be practical.

Speculative data dissemination is studied by [Bes96] using supply/demand controlled by the web server. An important parameter is that an object neighboring recently accessed object is likely to be accessed in the future. Communication cost, web server cost, stride cost (window size), session time (time of the cache), maximum size of disseminated object, policy to service an object, and history length of simulation estimation update cycle are used as input parameters. Web server load, service time, object miss rate at client, and amount of traffic increased are collected as outputs.

Heddaya et al. [HMY97] proposed diffusion-based caching protocol so as to balance web server load, but network throughput and client response time were also considered as performance parameters. Their simulation results by **WebWave** simulator were based on linear network topology and constant (10 seconds) gossip time. There was no explanation why they selected 10 second gossip time. It should be dynamic for general purpose applications. Since the Internet topology is not as simple as linear network topology; their work also should be extended to the other network topologies.

There are three types of push caching according to the number of received objects. First, Push-1 is defined as when the web server sends the object to only one of the eligible caches in the cache system. Second, Push-half is defined as when the web server sends the object to half of the eligible cache servers. Third, Push-all is defined as when the web server sends the object to the all eligible caches [TDVK98]. Deciding on which push caching may not be trivial work. In conclusion, Push-half should be extended to the Push-many where the web server sends the object to the number of caches varied by the time.

Another push cache method to decide which objects to push is finding a relationship between requested objects. The embedded dependency is defined as when $R_i$ and $R_j$ are two consecutive requests so that resource $R_j$ is embedded in resource $R_i$. The other one is traversal relationship when there is a hyper-link between those two consecutive requests [WS97]. **Wcol** retrieves certain number of image objects by looking at embedded image files in the requested object [wco, iCY97]. Since their approach pushes a certain number of embedded objects, it may not be beneficial for many applications. If clients are not interested in embedded images or interested in other embedded objects, this method may not be useful for improving web performance.

There are three types of dependency for computation those are single-pair, closure, and multi-pair. The single-pair dependency is defined as an access percentage of two consecutive accesses that exceeds a given threshold. The closure dependency is defined as a transition closure of the single-pair dependency between two resources. The multi-pair dependency is defined as one of the resources and all other resources that are accessed within a given window regardless of any specific order [WS97]. It may not be easy to decide window size, threshold for dependency graph. Finding a dependency graph for recursive structures is an important research topic. Finally, computation complexity is another performance bottleneck.

In summary, there are five groups of metrics for push caching – construction of the object dependency graph, the size of look ahead window size, the push threshold, the number of pushed objects, and dissemination method. Calculating those metrics during the pushing operation may not be easy and the computation overhead will be expensive as well. Thus, it is difficult to talk about its performance.

### 3.8.2 Client Initiated (Prefetch) Web Caching

The client requests can be categorized into three groups: wanderer, sojourner, and resident client [WS97, HWMS98]. First, wanderer client makes a few requests to any web server. For example, the client may be looking through a search engine. Second, sojourner client makes many requests, which are not focused on any particular web server, but the client is not bouncing quickly. Third, resident client makes many requests for a particular web server such as regarding material at the digital library. Selection of client type may not be trivial, but it could be static or dynamic depending upon the cache system implementation.

Using dependency relations at the client or the cache server [PM96] can do prefetching. Prefetching by using partial match predictions (by Markov model) is done by Palpanas and et al. [PM98]. When client requests are more dynamic than web objects, client initiated push cache keeps track of client activity to determine when the client's interest changes [DCW96]. The world is considered constant for the client initiated push cache, but the client's interest is dynamic. When client profile changes frequently, client initiated push caching may not be a good candidate for better web performance.

In short, client initiated push caching should optimize available network bandwidth, object popularity, object freshness, and object type. The prefetch mechanism simply parses the client request, retrieves the embedded (hyper-linked) objects, and stores them in the cache. Thus, the prefetch algorithm should use available network bandwidth efficiently. Traffic controllers keep track of object information such as URL, object type and size, distance metrics (distance from the web server to the cache), access frequency, and update frequency of the object [IKY97].

As a conclusion, client initiated push caching has more advantages for clients when client access patterns are dominant factors of web performance. New research studies should focus on client initiated push caching (prefetching) to achieve higher client satisfaction and better web performance. Prefetching method can be consisted of client, network, and web server resources to reach better performance. In the following chapter, our approach will be explored and discussed in detail.

### 3.8.3 Push vs. Prefetch Caching

Both server and client initiated web caching have advantages and disadvantages depending upon object and client request characteristics. When an object popularity is easy to know at the web server, server initiated push caching has advantages over client initiated push caching. If the client access profile is easy to determine, client initiated push (prefetching) has more advantages. However, predicting popularity of objects, clients, and web servers still remains an open research problem. Efficiency of the prediction algorithm, gaining of performance adaptability of the prediction algorithm, and easiness of implementation are design criteria of push caching. In addition to popularity of client and object, data dissemination is one of the important issues for obtaining higher web performance. Combining both server and client initiated push caching will be beneficial to both of them [HWMS98, WS97, MC98]. New research studies should focus on considering all web resource parameters to achieve higher and better web performance.

## 3.9 Overview of Object Delivery Mechanisms

In this section, some of the web object delivery systems for the web cache architecture will be discussed briefly.

Adaptive File Distribution Protocol (AFDP) [CK96] provides reliable, rate-controlled delivery of data to multiple clients and automatically determines the best transmission mode, according to the number of clients, their capabilities, and support from connecting networks. The slowest client determines the transmission rate. If one of the packets is lost at any time for any one of the clients, all the packets are sent to all clients. Content distribution is organized by a secretary, which accepts object requests from clients and informs the web servers. Assigning a secretary and deciding whether multicast or unicast is an open research topic.

Continuous multicast distribution (CMP) is introduced for randomly updated web objects by Rodriguez et at. [RRB98]. It has five components. First, session servers or mechanisms are needed to map the name of the object into multicast addresses. Second, the web server needs to monitor which objects to multicast and when to stop multicasting according to the number of object requests and object change rate. Third, multicast capable routers introduce delays to maintain state information of each multicast group. Fourth, depending on the location of receivers, there is an overhead because of joining and pruning messages. Fifth, multicast congestion control is an open research topic. Non-popular objects should not be sent via CMP because of wasting network to maintain many multicast trees.

**Wormhole** provides a pipe with relatively constant latency between two points on the globe. INTELSTAT Internet Delivery System (IDS) [CAJ+99] has two levels which are called Warehouse and Kiosk. The transparent web caching system is used at Kiosks. Warehouse collects clients' web access statistics and decides popularity of objects by using client access history at all of the Kiosks. They proposed HTTP PUSH for multicasting objects in the cache system that provides content distribution to a group of caches in the protocol level. Extendibility of HTTP protocol will be discussed shortly. Dividing the Internet into two levels (Warehouse and Kiosk) may not be a good solution when scalability of network and network cost parameters vary.

Instead of sending one HTTP request for each web request, a combination of multiple HTTP requests has the opportunity for achieving higher web performance if the network conditions are reliable [CAJ+99]. Improving communication protocol can improve web performance as well. Chen et al. proposed HTTP PUSH method [CAJ+99]. When the network is stable and message drop offs are rare, it achieves higher web throughput. However, it may not be beneficial in case of high message drop offs at unbalanced and unreliable networks.

## 3.10 Categorization of Web Caching

Brian Davison surveyed proxy web cache evaluation methodologies based on source of workload and form of algorithm implementations [Dav99].

There are two types of web caching according to maintenance of an object consistency: passive and active caching. There is no object freshness mechanism unless the object requested by the client is in passive web caching. When the client requests an object, object retrieval time may take longer because of confirming availability of the object. When the object changes frequently, verifying the latest version of the object is a bottleneck.

Objects in the cache are asynchronously controlled during the peak-off network traffic times whether they are fresh or stale for active web caching. Therefore, it does not slow down network traffic during the client request. When the object changes rarely, active caching wastes available network bandwidth.

Although active caching delivers up-to-date (fresh) objects, it wastes network bandwidth. On the other hand, passive caching may have delayed responses to client requests because of confirming object freshness. Performance of web caching depends upon how often cached objects change and how quickly object validity is confirmed.

The best solution will be a combination of both passive and active to get advantages of both. Therefore, request frequency of the cached object, update frequency of the object, available network bandwidth characteristics between the web servers and the cache servers are main factors for higher web performance. To knowledge of the author and his advisor, there has been no such research seen in the literature so far.

## 3.11 Relation Between Related Work and Our Approach

When client and object access profiles change frequently and future client requests do not depend upon previous access patterns of clients and objects, previously discussed methods may not be a good solution. This work addresses that making a prediction for a group of users working under the guidance of a leader has the potential of achieving higher prediction and web performance.

Retrieving predicted objects in advance while utilizing network resources more effectively has advantages compared to the other prefetching techniques. Since predicted objects are brought to the proxy cache while current client requests are processing, conventional prefetching methods consume limited available web resources. This work addresses predicting and retrieving prefetched objects while utilizing web resources more effectively. Thus, the waiting time of the clients will be reduced and utilization of web resources will be much better than other techniques.

Client web requests can be done in anytime (random). Conventional web requests are not deterministic in time. Some web requests can be expected (deterministic) such as planned web surf activities. Predicting near future web requests requires additional information to predict prefetched objects.

Having planned web activity is not possible with current techniques. Having a web surf plan will help to use web resources more effectively. Thus, it will reduce latency and lead to better web performance.

Better web performance is considered as the least client waiting time (latency) because latency consists of all web resources and the purpose of the web is to deliver the objects to the clients with high client satisfaction. The other performance measures (e.g., web server load, network traffic utilization, etc.) may not be directly related to the client satisfaction.

The nature of the web is a demand driven mechanism. A client makes a web request and waits to receive a response from the web server or cache server. Some web requests can be done in advance if those requests are somehow predicted.

Since web resources are lightly consumed during peak off times, the web resources are not balanced throughout a day, week, month, etc. Balancing of web resource utilization leads to better and more cost effective web performance.

Any one of the web requests cannot be done and completed in advance with current web interactions such as bring tomorrow's newspaper today when the objects are not available at the moment the request is made. However, sending some web requests in advance gives flexibility to the client. It also gives an opportunity to utilize web resources more effectively.

As a conclusion, distributing web documents while utilizing web resources more effectively will be new direction and novel approach to reach higher web performance and better resource utilization. Contribution of this work may change content distribution between clients and servers in general.

Building interest groups in the simple cache and throughout the cache hierarchy has the advantage of predicting future web requests. Delivering web objects to the interest groups in the simple cache and the entire cache hierarchy will improve web performance by using web resources more effectively.

Building interest group hierarchy via cache hierarchy will take advantage of a group of clients and objects, and proxies to use web resources more effectively.

Even if using only one cache hierarchy, a larger storage capacity is needed for the higher levels in the cache. If the interest groups objects are built on a hierarchy vs. cache hierarchy, data storage and exchange mechanism will be distributed.

When web resources and client requests are well distributed in time and place, web performance will be much better.

As a conclusion, better web performance from the client's point of view will give better client satisfaction. Outcomes of this work may change data dissemination among data sources and requesters throughout the connection system.

## 3.12 Discussion

Since previous studies and implementations have some disadvantages which were discussed previously, new studies fill some of the voids in this field.

Sharing objects with others via the proxy cache and predicting future web requests will potentially increase web performance and client satisfaction. By contrast with other prefetching methods, this work addresses a novel prefetching mechanism of bringing predicted objects while web resources are lightly used. The instructor makes a prediction for a group of clients while they work under the guidance of the instructor.

Web performance will be much better by prefetching those predicted objects to the local proxy while web resources are lightly used. Our approach which is **The instructor initiated prefetching** is explained in detail in the following chapter.

This work is useful for a group of clients and individual usage when each client has a planned web activity. Since web requests are done in advance, waiting time for the client is much smaller than direct connection.

Building groups, which have the same interests, cooperating with other groups and coordination via the cache hierarchy by prefetching objects in advance while utilizing web resources which are lightly used, will have strong impact on better web performance.

## 3.13 Conclusion

Since prefetching techniques give opportunities for achieving better web performance, recent research studies are moving to predict future web requests and distributing web objects. Although most of the prefetching techniques have been using previous client and object access patterns, some web requests can be predicted with high efficiency in several cases. Instead of predicting future web requests for all clients, having better web performance for a group of clients during a certain time interval will achieve much better web performance. This approach is novel to the web caching and web philosophy. Because most of the prediction techniques (prefetch and push web caching) predict near future web objects while current the client web request is processing, their web resource utilization may not be effective. On the other hand, some web objects may be predicted with high efficiency and brought to the cache while network resources are lightly consumed or even not being used. This approach not only improves web resource utilization but also achieves better web performance.

# Chapter 4

# The Instructor Initiated Prefetching

*Today's children, tomorrow's leaders*

*One tiny step is the beginning of the following steps*

*Listen a hundred times; ponder a thousand times; speak once*

In this chapter, the instructor initiated prefetching is defined. Then, the model of the instructor initiated prefetching is introduced. Implementation of the instructor initiated prefetching is introduced and experimental information is given. First, implementation details of the instructor initiated prefetching model are explained. Cache statistics for the instructor initiated prefetching are explained briefly later. Then, one complete example with all prefetch related details is explained and discussed. **Squid** proxy server [squ] selection reasons are summarized. Later, squid and new cache replacement policies are discussed. Next, prefetch related options added to the squid proxy server (v.2.2) will be introduced. Cache server features are briefly introduced. Outcomes of this method are compared with others. Distinctions of this method from other studies are discussed. Contributions of this method are summarized. Suggested future work is given briefly. Finally, discussion and conclusion of this chapter are given.

## 4.1 The Instructor Initiated Web Caching

### Why a School Environment is Chosen?

The origin of Internet clients can be categorized into four groups. Those are clients from home, work, school, and others. Clients from home could be anyone working on the Internet. Clients from work could have various purposes and expectations. The last group is other type of clients. Since there is no way to characterize those clients, making a distinction from others is almost impossible. It is hard to make a large interest group or groups because of the variety of work, home, and other environments. There is a slight possibility of making small interest groups of clients who make connections from work. Clients from a school could have a main goal, which is to retrieve web objects for educational purposes. Making interest groups for the education environment has more advantages than the others. When each student has access to the Internet, the number of clients goes higher as popularity of the web among students becomes significant. In addition, the number of schools connected to the Internet has been increased so that percentages of schools connected to the Internet from 1994 through 1998 are 35, 50, 65, 78, 89, respectively [Library Resources and Technology: Libraries, page 477]. According to those statistics, spending resources more intelligently improves web performance (client, web server, and network) and spreads out the Internet usage in the long run in many fields more efficiently. As a result, the school environment is well suited for building interest groups.

Let us consider an example; a class of students searches information for 'dolphins' on the web. Each one of the students will probably use one of the web search engines such as yahooligans, dogpile, excite, altavista, infoseek, etc. They will probably get the same web objects since they use the same keywords and search engines. In addition, each one of the students creates almost the same web network traffic and object requests to retrieve the

Figure 4.1: Instructor initiated prefetching (web caching) model

same objects during the same class period. Most of the K-12 schools have very limited network bandwidth and require broad network bandwidth during the school times.

A group of students with an instructor has a schedule to work in the computer lab to do a search on the web. Since the instructor and his/her students need to follow the curriculum, their web object requests will most likely have a significant overlap for a certain time period. If the instructor visits the possible web objects through the cache server before students request them, students get the objects from the cache server instead of retrieving those objects from the original servers. When a conventional cache server is used, web objects are cached while students are making their web requests during the computer lab hours. Expecting highly loaded network traffic and web server utilization during the lab hours, the client waiting time will be longer than that of peak off times. Even using conventional prefetching techniques discussed in the previous chapter may not give better web performance since they use previous web requests and objects to predict near future client requests and retrieve those objects while the current client request is processing. Since the instructor requests web objects in advance, this method could be called **Instructor Initiated Prefetching/Web Caching**.

## 4.2 Suggested Model

The model gets web cache filling parameters from the instructor such as search topics or keywords, class meeting time, possible/preferred web sites, and optional features. The model collects candidate web object pointers according to the instructor's input and visits those candidate objects through the cache server while the network is lightly loaded.

### Which Objects Need to be Prefetched?

Although some conventional web prefetching methods predict client requests by using current or previous client access patterns, the instructor can do better web object request prediction. When students and the instructor have a curriculum to follow, their web requests are expected to highly overlap any given topic during a particular time interval. When they use the same search topic or keywords at certain search engines or web sites, their web object requests are more or less the same. This new method uses prefetching for a group of clients (students) working under guidance of the instructor. On the other hand, conventional prefetching methods make predictions for all clients. Students who are guided by the instructor will get the benefit of prefetching, however the other clients may not see improvement of web performance explicitly. In addition, reducing web traffic between the clients and the Internet, and reducing some of web servers' load will give a better web performance from the client's point of view even though some clients are not part of the interest group..

Prefetching would be better because future client requests are predicted by the instructor. When students do not follow the directions of the instructor, this method may not give high web performance. Clients still have benefits of using the proxy server. When client requests are highly overlapped with the interest of the instructor and the number of clients goes higher, web performance will be much better. Then, latency depends upon object retrieval time between clients and the proxy.

### When Do We Fill Out the Cache?

The local cache needs to be filled out before any one of the students' request any web object. This time could be between after the instructor makes the prefetched object request decision, and right before the class starts. One of the advantages of the instructor initiated web caching method over conventional methods is that prefetched objects are brought to the local cache during the light network traffic times, e.g. local night times, weekends, holidays, etc. The other prefetching methods retrieve web objects while current client request is processing.

### How Do We Maintain the Cache Server?

Since the local disk storage capacity is limited, some objects need to be replaced when the local cache is full to allocate a new object. Some previous research studies have been done to evaluate a variety of cache replacement policies [KKO98, DAP99, ARA$^+$95, WAAF96]. In those studies, web performance also depends upon how the cache replacement policy is implemented. Since instructor initiated prefetched objects have a greater chance of being requested in the near future (during the class period) than other objects, the cache replacement policy should give higher priority to those prefetched objects until the class period is over. As soon as the class period is over, those prefetched objects have to be removed immediately from the local cache.

As a conclusion, the cache replacement policy for the instructor initiated web caching (prefetching) has to make sure that prefetched objects have the highest priority during the class period to stay in the local cache and have to be removed as soon as the class is over. Having multiple cache replacement choices for different web objects leads to better local cache storage utilization. Thus, the better web cache employment achieves the better web performance.

## 4.2.1 How to Measure Web Performance and Collect Statistics

Web performance is explained in detail in the following chapter. The main goal of this work is reducing the waiting time of clients. Thus, web performance from the client's point of view is considered. Proxy cache shows client service time from the proxy's point of view. Client processing time, object hit, object byte hit, prefetched object hit, and prefetched object byte hit are collected via the proxy server. Prefetch and object hit ratios and prefetch factors are calculated. Although some results show web and cache performance as hit rate, byte hit rate, and average/medium user service time or latency, their results are a collection of statistics based on their experimentation setup. Since they rarely specify their setup and collection method, their results (e.g., efficiency, improvement, etc.) may not be enough to come a conclusion. On the other hand, this work shows web performance, measurement method, and collection of data both in fundamental level and good enough to come to a conclusion. Thus, the web performance has to be defined carefully, analyzed thoroughly, implemented intelligently. The results have to be collected meaningfully, and judged fairly under the circumstances and in general.

## 4.3 Implementation of the Instructor Initiated Prefetching

The instructor initiated prefetching model is discussed above in detail. Implementation details of the instructor initiated prefetching are introduced in this part. In short, some of the objects (documents) need to be requested in advance while network traffic is lightly loaded. Then, those documents are found in the cache when a group of users who follow the instructor makes those requests. This method will reduce latency for highly possible web object requests in a particular time interval especially clients working under the narrow network bandwidth restrictions.

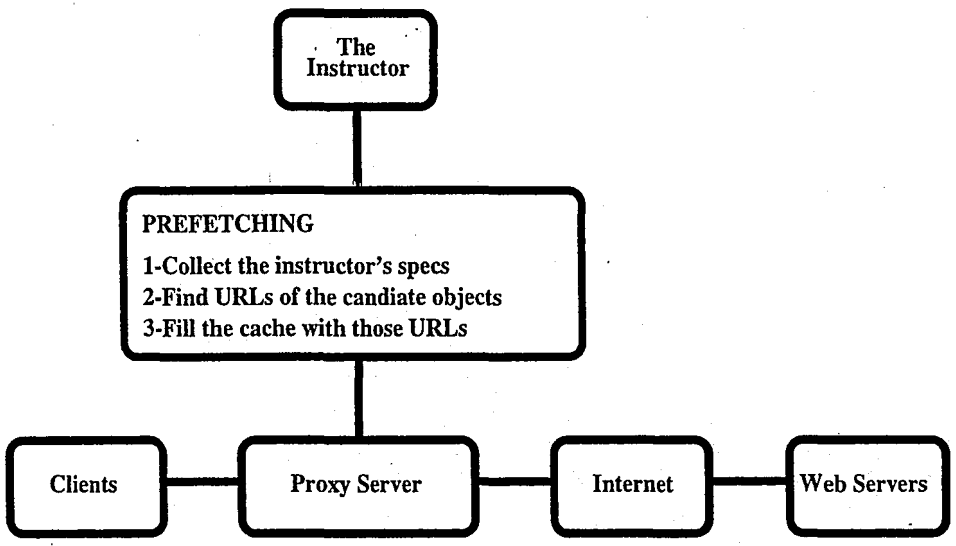
The Instructor
PREFETCHING
1-Collect the instructor's specs
2-Find URLs of the candiate objects
3-Fill the cache with those URLs
Clients
Proxy Server
Internet
Web Servers

Figure 4.2: Implementation of the instructor initiated prefetching

Programs are developed to implement the instructor initiated prefetching. Each client request goes to the Internet through the proxy server. The instructor communicates with the proxy server via the prefetching module. The class has scheduled work in the computer lab to surf the Internet on a particular topic. The instructor knows when and what highly possible web objects will be requested by the class. Thus, the instructor makes a prefetching query to bring some possible objects to the cache before the class starts. When the class makes the search on the same topic, some objects are found in the cache. The effectiveness of this method depends upon how much the class requests and prefetched objects are overlapped.

The instructor initiated prefetched model is denoted by **Prefetching** module in Figure 4.2. The prefetching module has three parts:

1.Collect the instructor specs

2.Crawl/Find candidate URLs according to the instructor inputs

3.Fill the cache with those URLs during night through the cache server

**Submit Instructor Specs:**

First, the instructor specs have to be given to the prefetching system. Those specs can be given either via the web or modifying the appropriate file/s. The instructor should submit prefetching details here such as search keywords, preferred search engines, preferred sites to follow, and preferences.

The instructor submits the following information to prefetch possible objects by accessing an online form at:

`http://severname/inst/cache_request.html`

Servernames are:

`vergence-i.eecs.lehigh.edu` for VAST lab and,

`204.170.130.23` for Broughal Middle School.

The instructor should enter search keywords in the empty field.

Search engines are metacrawler, google, hotbot, altavista, yahoo, yahooligans, ajkids. Metacrawler is chosen as a default search engine in case the instructor forgets to choose one. In addition, either one of the search engines can be selected as a default search engine.

Preferred sites must be given in each line with a complete URL. Those sites will be automatically added to output file of the crawler program (URL.txt). The instructor should submit the complete URL of the prefetched object in each line if there are some preferred sites to follow.

Preferences follow with their default values:

Returned number of hits (RETURNEDHITS) is 30 as a default. It has not been used with the current version of the program.

Time to wait for replies from each search engine or the direct URL (TIMEOUT) is 30 seconds as a default.

Maximum number of minutes to spend on the task (prefetching) (MAXTIME) should be set at 60 minutes. Its default value is 30*60 seconds (30 minutes).

Crawl through the same site first (TRYOCLOSEURLSFIRST) is true as a default. It has not been used in the current version of the program yet.

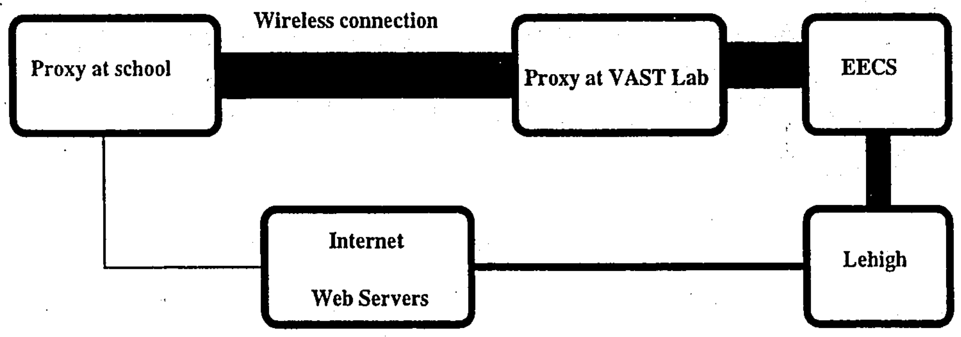

Figure 4.3: Block diagram of cache hierarchy system

Recursion depth (RECURSEDEPTH) is assigned 2 as a default. It will be used while the filling cache program is running to fill the cache. It defines prefetching detail and cost of prefetching. It should be defined carefully.

The number of HTTP connections to open at once (MAXCONNECTIONS) is assigned 4 as a default. It is related to the crawler program.

Maximum number of pages to cache (MAXSITES) is assigned 100 as a default.

Modifying source code of the crawler program package can change those default option values. Default option values can also be modified by updating `cache_request.html` web page file.

Network traffic at Broughal Middle School was observed after installation of the proxy server. There was no traffic coming through the proxy server from any one of the users at the school during off-school times (between 4 PM to 8AM during the school days and holidays). The proxy server was installed at the school in summer 1999. The other proxy server was installed in VAST lab at the same time. Simple cache hierarchy was built and dedicated wireless communication was established between proxy servers.

The instructor has to submit prefetching specs by the first day of the class at 2AM. Prefetching time is chosen between 2-3AM since network traffic at the school is lightly loaded during that time. This prefetching time can be changed by modifying the proxy server configuration file, which is usually located at `PROXY_PATH/etc/squid.conf`.

All the instructor preferences and prefetching options are written into a file called 'URL.txt'.

**Crawler:**

Second, search queries are sent to search engines according to the instructor input and search returns will be collected as a full URL for each candidate object. The crawler program takes the instructor's inputs, and then it sends search requests to selected search engines on behalf of the instructor. Returned object URLs and prefetching options are written into a file called 'URL.txt'.

**Fill Cache:**

Third, those candidate objects should be brought to the local cache before the class starts. Fill cache program sends HTTP requests with a candidate URL found by the crawler program to the web through the cache server. The cache filling operation is done between 2-3 AM of the first day the class that needs instructor initiated prefetched objects.

All program packages for the instructor initiated prefetching are completed to implement experiments. Second and third part of the prefetching modules run during the prefetching time (e.g. 2-3AM). The instructor specs have to be given before 2AM.

### 4.3.1 Why Squid Proxy Server is Chosen?

Among the most widely used proxy cache servers today is **Squid**. It has an open source software package `http://www.squid-cache.org` [squ]. It is highly portable, freely available, and has an active and experienced developer community that provides free support. Many people who are well known in the web caching community have been working on this package. Many research results used squid proxy server in their papers. Thus, some results may be compared with results of this work if it is necessary. It also allows building a cache hierarchy.

## 4.3.2 Addition to New Squid Configuration Options

A new configuration option is added to the squid source code to give flexibility to the instructor for how long prefetched objects are needed. It is called precache_reference_age. Its default value is 1 day. Instructor initiated prefetched objects stay in the cache at least 1*7 days with a higher chance than other objects. If there is no request after 7 days, those objects are removed faster than other objects. This option specifies how long prefetched objects are needed and how fast prefetched objects have to be removed from the cache.

A new cache server configuration feature is added to specify prefetching time. This time can be given a one hour period starting with the number (e.g. 0, 1, 2, ..., 23). The default prefetching time is assigned with precached_hour as 2 (2AM).

Any one of the squid configuration options can be updated by modifying the **Squid** proxy configuration file which is usually located at `PROXY_PATH/etc/squid.conf`.

## 4.3.3 Collected Cache Statistics

In this section, prefetching related cache statistics are explained. Those statistics define how well caching and prefetching contribute to web performance.

**Definition of object hits and prefetched object hits**

Object hit = Summation of all the counters of appropriate object hits ($TCP\_HIT + TCP\_REFRESH\_HIT + TCP\_MEM\_HIT + TCP\_IMS\_HIT$)

For more information, refer to the squid proxy server documentation or source code in [squ].

Prefetched object hit =

Summation of all the counters of the appropriate prefetched object hits =

$$(TCP\_PRECACHE\_HIT + TCP\_REF\_PRECACHE\_HIT + TCP\_MEM\_PRECACHE\_HIT + TCP\_IMS\_PRECACHE\_HIT)$$

Further information can be obtained by looking at the developed source code for this work. Contact either the author or the advisor.

After employing the web cache server, one may be curious as to how web performance was effected. The squid proxy cache statistics are generated by the client on demand.

Statistics related to the instructor initiated prefetching can be seen under 'precache special' of the cache manager web page at:

`http://servername/cgi-bin/cachemgr.cgi`

In this part, instructor initiated prefetching statistics are explained briefly with their variable name between parenthesis.

http_requests (client_http.requests): The number of client requests which came to the proxy server.

http_hits (client_http.hits): The number of objects requested by clients found in the cache server other than prefetched objects.

prefetch_hits (client_http.precache_hits): The number of instructor initiated prefetched objects requested by clients found at the cache server.

Request_hit_ratio: The ratio of the objects (not including prefetched) found in the cache to the total number of object requests.

Request_hit_ratio = http_hits / http_requests

Prefetch_hit_ratio: The ratio of the prefetched objects to the total number of object requests.

Prefetch_hit_ratio = prefetched_hits / http_requests

Prefetch_hit_factor: The ratio of the prefetched objects to the total number of objects found in the cache (prefetched and hit objects).

Prefetch_hit_factor = prefetched_hits / (http_hits + prefetched_hits)

http_kbytes_out: The total number of kbytes (thousand bytes) sent to clients.

hit_kbytes_out (client_http.hit_kbytes_out): The total number of kbytes sent to clients from the cache and not including that of prefetched objects.

prefetch_hit_kbytes_out (client_http.precache_hit_kbytes_out): The total number of prefetched kbytes sent to clients.

Byte_hit_ratio: The ratio of the total size of the objects (not including prefetched) found in the cache to the total number of Kbytes sent to clients.

Byte_hit_ratio = hit_kbytes_out / http_kbytes_out

Prefetch_byte_hit_ratio: The ratio of the total size of the prefetched objects to the total number of Kbytes sent to clients.

Prefetch_byte_hit_ratio = prefetch_hit_kbytes_out / http_kbytes_out

Prefetch_byte_hit_factor: The ratio of the total size of the prefetched objects to the total size of the objects found in the cache.

$Prefetch\_byte\_hit\_factor = prefetch\_hit\_kbytes\_out/(hit\_kbytes\_out + prefetch\_hit\_kbytes\_o$

All requests and hits are positive integer numbers and represents counts. All ratios and factors are in percent (%). All object sizes are in kbytes (thousand bytes). Accessing the 'Precache Special' link under the cache manager menu from the cache statistics web page shows all the statistics, however all factors need to be computed separately. It is also possible to see each statistic for each individual client (identified by a distinct IP address).

### 4.3.4 Cache Replacement Policy

Several cache replacement policies for proxy caches are summarized in the previous chapter. In this section, two cache replacement policies are explained briefly. The first one was used in the squid proxy cache (for version 2.2) [squ] and squid cache replacement policy (aging function) is called LRU_age (Least Recently Used). This cache replacement policy removes least recently requested objects from the cache to make enough room for the new coming object. The second one was used for the instructor initiated prefetched objects and it is called LRU_precache_age. It keeps prefetched objects for a certain time in the cache with high priority and removes prefetched objects faster than others when no one requested the object for a certain time.

LRU_age algorithm computes the age factor of the current cache utilization (swap_factor). Then, any object which has not been requested longer than this age will be removed from the cache.

```
LRU_age()
{
  x = swap_factor
  age = 60*pow(Config.referenceAge / 60, x) = 60*pow(43200, x)
  if age < 60 then age = 60
  if age > 31536000 (1 year) then age = 31536000
  return age
}
```

$x = swap\_factor$

$= (store\_swap\_high - store\_swap\_size)/(store\_swap\_high - store\_swap\_low)$

$= (95 - store\_swap\_size)/(95 - 90)$

$= (95 - store\_swap\_size)/5$

$x$ is also between 0 and 1 ( if $x < 0$ then $x = 0$; if $x > 1$ then $x = 1$).

Config.referenceAge = 1 month = 30 * 24 * 60 * 60 = 2592000 seconds.

pow(x, y) = x raised to the y.

Age function of LRU = LRU_age = 60 * pow(43200, x)

swap_factor represents how quickly local disk space is needed. When the store swap size closes to the store_swap_high (95), exponential term (x) becomes a small number (close to 0), and age becomes small. Thus, objects that have not been requested at least

LRU_age times or longer have to removed from the local cache quickly. When store swap size closes to store_swap_low (90), exponential term becomes close to 1. LRU_age function returns a high number compared to the previous case. Then, the objects have a high chance of staying in the local cache.

Base of the exponential function of the age depends upon the Config.referenceAge parameter which can be chosen in the squid proxy server configuration file

`PROXY_PATH/etc/squid.config.`

Config.referenceAge is the dominant factor of the age function. It determines how long objects need to stay in the cache. If the object has not been accessed for age time (1 month according to our configuration), it is removed from the local cache immediately when store_swap_size reaches the upper limit (store_swap_high).

In order to implement the instructor initiated prefetching, the cache replacement policy should be modified. Our goal for the instructor initiated prefetching is to bring some objects, which are highly likely to be requested in the near future. Those objects need to stay in the cache for a certain amount of time. Then, they need to be removed faster than other objects after the scheduled time is passed. In our school example, the instructor requests the objects for one day in advance before the class experiment starts on the web. Some of the objects may be used in similar classes or the same objects may be taught for another group of students. Then, those objects have to stay in the cache at least a week. If any precached object is not requested for a week, then those prefetched (the same as precached) objects need to be removed from the local cache quickly.

The new algorithm LRU_precache_age is given below:

```
LRU_precache_age(time not_requested)
{
  x = swap_factor
  if (not_requested < 7*Config.precache_referenceAge = 604800)
```

```
      z = pow(config.precache_referenceAge, x) = pow(86400, x)
   else
      z = pow(config.precache_referenceAge/60, x) = pow(1440, x)
   age = 60*z
   if age < 60 then age = 60
   if age > 31536000 (1 year) then age = 31536000
   return age
}
```

config.precache_referenceAge = 1 day = 24 * 60 * 60 = 86400

Age function of precache LRU = LRU_precache_age :

if $not\_request < 604800$ then $60 * pow(86400, x)$,

if $not\_request > 604800$ then $60 * pow(1440, x)$

Swap_factor, x, is the same as used in the previous algorithm. Some other classes may use those prefetched objects. Two aging functions are used in this experiment to decide depending upon whether the instructor initiated prefetched object or other object. If prefetched documents have been requested less than a week, they should stay in the cache. If prefetched documents have not been requested for more than a week, those objects have to be removed from the cache quickly. LRU_precache_age is only applied for prefetched objects.

In conclusion, a new cache replacement algorithm for the instructor initiated prefetching assures that prefetched objects have to stay in the cache with a higher chance than other objects during the precached time period. As soon as the precached time (scheduled working time) is passed, those precached objects have to be removed faster than other objects. The squid cache replacement policy (LRU) is used for all non-prefetched objects.

### 4.3.5 The Instructor Initiated Prefetching Experiment

All the residual objects in the cache server were removed after the local cache storage was cleared. The proxy server at the school was restarted on June 8, 2000 for fresh run before the instructor initiated prefetching experiment was started.

Our experiment class was going to surf on the web to get information about 'volcano'. The instructor suggested using 'ajkids' and 'yahooligans' search engines to find related objects and web pages. The class was supposed to work on June 9, 2000 between 10AM and 12PM (noon) at the computer lab. We want to prefetch some possible web objects/pages according to the instructor initiated prefetching model explained above.

The instructor gave the following prefetching information on June 8, 2000 at 4PM. Prefetched documents were needed by June 9, 2000 at 10AM. The following instructor inputs were entered at the school cache server:

Search keyword = volcano

Search engines = ajkids and yahooligans

Maximum number of minutes to spend on this task = 60 minutes

The other preferences were used with their default values explained previously.

In this experiment, crawler sent search requests to 'ajkids' and 'yahooligans' to search objects for 'volcano'. Then, crawler collected a list of URLs from the returned search results. Those URLs were written into 'URL.txt' file. The crawler program can be run anytime before the filling cache program started (June 9, 2000 at 2AM). Then, the web cache server at the school was filled with those URLs found by the crawler program between 2AM and 3AM on June 9, 2000.

Prefetching statistics were collected in addition to default statistics of the squid cache server. Cache server statistics can be seen at:

`http://204.170.130.23/cgi-bin/cachemgr.cgi`

| TIME EST (+4 for GMT) | 3:03 | 13:07 | 16:25 |
|---|---|---|---|
| http_requests | 540 | 2979 | 3414 |
| http_hits | 50 | 1241 | 1326 |
| prefetch_hits | 13 | 146 | 146 |
| Request_hit_ratio | 9.25 | 41.65 | 38.84 |
| Prefetch_hit_ratio | 2.40 | 4.90 | 4.27 |
| Prefetch_hit_factor | 20.63 | 10.53 | 9.92 |
| http_kbytes_out | 4761 | 19766 | 21760 |
| hit_kbytes_out | 162 | 4818 | 5036 |
| prefetch_hit_kbytes_out | 344 | 1078 | 1078 |
| Byte_hit_ratio | 3.40 | 24.38 | 23.14 |
| Prefetch_byte_hit_ratio | 7.23 | 5.45 | 4.95 |
| Prefetch_byte_hit_factor | 67.98 | 18.28 | 17.63 |

Table 4.1: Instructor initiated prefetching results were collected in June 9, 2000; search keyword = volcano.

Cache host, and port number of the web cache server were entered as localhost and 2001 respectively. Manager name and password fields were left empty because of squid cache server configuration at that moment.

Cache manager menu with a list of links allows us to obtain various cache statistics. Then, 'Precache Special' should be selected to see prefetching statistics. The instructor initiated prefetching statistics are shown in Table 4.1. Those results were gathered from the proxy server at the school.

All the requests and hits are integer numbers that represent the total number of requests or hits. All ratios and factors are denoted as a percentile (%). All kbytes_out represent sum of transferred object size from the cache server in kilo (thousand) byte units.

The second column of Table 4.1 represents the cache statistics as soon as prefetching process was completed. After prefetching was done, some of the objects were requested to see whether or not prefetched. Therefore, there were hits and prefetched hits collected even though the school was closed at 3:03AM. The number of prefetched objects is 477

(540 - 50 - 13). Size of the total prefetched objects would be 4255 (4761 - 162 - 344) Kbytes.

The third column shows the cache statistics after the experiment class, which used the instructor initiated prefetching objects, resumed. We were told that some students might have computer lab even though most classes resume at 3PM.

The fourth column shows the cache statistics for end of the first school day of the experiment for the instructor initiated prefetching. All the classes and labs are completed by 4PM.

**Analysis of Table 4.1:**

According to the third column, the instructor initiated prefetching hit factor (prefetch_hit_factor). 10.53 % of the total hits (41.65 + 4.90 = 46.55%) is supported by the prefetched objects. In another words, 10.53% of the objects were found in the cache that was brought in advance. Prefetch_hit_ratio is an additional hit ratio which contributed by the instructor initiated objects. However, 18.28% of the total hit object size (24.38 + 5.45 = 29.83%) came from prefetched objects. This result is much better than the hit ratio factor.

The fourth column represents the cache server statistics at the end of the experiment school day. In fact, there was no prefetched object requests after 13:07. It means that no one requested instructor initiated prefetched objects. We also know that the experiment class was in the morning. Since the cache server had been used by others, there are some hits for them as well in the fourth column.

Even though the prefetch request hit factor was around 10%, transferred prefetch byte hit factor was around 18%. There is a need to determine cache (implicitly web) performance with this new approach. Web performance is explored in the following chapter.

Prefetched http request ratios for each client went up from 0 to 19%. It means that some of the clients (students) followed the instructor better than other clients [See Appendix]. Even though the instructor told the class to follow the instructor, some students may not follow well.

A total of 146 prefetched objects were requested while 477 objects were brought to the cache in advance, which was a good turn out. 30% of prefetched objects were requested. 1078 Kbytes of prefetched objects were requested out of the total prefetched objects (4255 Kbytes). All prefetched objects stayed in the cache during the experiment period (June 9-20, 2001).

**Results:**

- The instructor initiated prefetching helped hit ratio.

- The prefetched objects were requested during the scheduled time. As soon as the scheduled time was over, prefetched objects were rarely requested as expected.

- Byte hit factor was higher than the request hit factor of prefetched objects. Thus, latency improvement would be better because size of the objects is one of the dominant factors on object retrieval time.

- Some of the clients followed the instructor better than others. Prefetched hit ratio of some clients went up to 19%.

**Discussion:**

- Since the cache server had been used for other students who did not follow the instructor, percentage of hit and size improvement of prefetched objects may not reflect real improvement after the experiment class resumed.

- Since we did not have a chance to repeat the experiment with different instructor preferences, changing some of instructor parameters may effect prefetch improvement (e.g., recursion depth, search engines, etc.).

- Even though students should follow the instructor guidelines, some of them may surf on the web freely.

- There is no enforcement for the students to obey the instructor.

- The model (cache, or web) performance should be measured for a particular time interval.

| DATE<br>TIME EST (+4 for GMT) | June 14, 2000<br>01:25 | June 20, 2000<br>12:08 |
|---|---|---|
| http_requests | 10135 | 10369 |
| http_hits | 4397 | 4463 |
| prefetch_hits | 148 | 148 |
| Request_hit_ratio | 43.38 | 43.04 |
| Prefetch_hit_ratio | 1.46 | 1.43 |
| Prefetch_hit_factor | 3.26 | 3.20 |
| http_kbytes_out | 74421 | 75633 |
| hit_kbytes_out | 24155 | 24230 |
| prefetch_hit_kbytes_out | 1082 | 1082 |
| Byte_hit_ratio | 32.46 | 3.20 |
| Prefetch_byte_hit_ratio | 1.45 | 1.43 |
| Prefetch_byte hit_factor | 4.29 | 4.27 |

Table 4.2: Instructor initiated prefetching results collected in June 14 and 20, 2000.

**Analysis of Table 4.2:**

The results of Table 4.2 contained prefetched statistics after the instructor initiated prefetching was done on June 9, 2000.

Prefetch hits are changed from 146 in Table 4.1 to 148 in Table 4.2. Instructor initiated objects are used mostly on the first day of the experiment. During the June 9-20, 2000, there were only 2 prefetched object requests. Thus, instructor initiated prefetched objects were requested while the class was working under the instructor. As soon as the class was finished, prefetched objects need to be removed from the cache. The new cache replacement algorithm works exactly as expected. The results as shown in Table 4.1 and 4.2 support the new cache replacement algorithm.

**Some Extensions to Cache/Web Performance Statistics**

Additional collected cache performance statistics related to prefetching can be found at the Appendix at the end.

- Median service time in seconds for recent 5 minutes and 60 minutes for (cache hits, precache hits, refresh hits, refresh precache hits, not modified replies, not modified replies for precache) are collected.

- Those data measurements have some meaning but not enough to express latency from the user point of view.

1.It gives median service time only last 5 or last 60 minutes.

2.It considers all requests, all clients (some may not be a group member).

- Novel web performance measures should be defined and implemented (some of them will be explored in the web performance part).

- Detailed web performance measures should be done to separate and include web performance related issues (e.g. proxy overhead, local cache properties, request handling, cache replacement policy, object retrieval and storage issues at the local cache, definition of latency (object retrieval time, client waiting time, service time of (1, progressive, or all object bytes))).

- Detailed measurements and representation of statistics will be beneficial for object, client, web server, groups, time interval, etc. to see which factors have impact on web performance.

**Experimental Results:**

- Instructor initiated prefetched objects were mostly requested during the scheduled working time period. Prefetched objects were rarely requested after this time period passed.

- Conventional cache performance measures supported by squid pro~~xy~~-server and additional prefetching measures were done such as total object and prefetched object hits, object size and prefetched object size, medium service time of object hits and prefetched object hits for a time interval (last 5, 60 minutes). Many more cache statistics could have been derived from the proxy log.

- Object hit factor, byte hit factor express the contribution of prefetching.

- Prefetched object request ratios for each client varies from 0 to 6 % since there was only 2 prefetched object requests after the first day of the experiment [See Appendix].

Therefore, the prefetch hit ratio throughout the total run time of the proxy server was dropped dramatically for the following days.

- Cache replacement policy was implemented for prefetched and non-prefetched objects since their usage is different.

**Comments:**

- A small experiment is the starting point. It gives the importance of the methodology and philosophy of meaning to the problem and solution.

- This opens many fields to explore as explained throughout the work. One needs to read carefully and be patient to understand the whole writing.

- Comparison of this work is how much similar data or enough data to express the importance of the work.

- Comparison of the experiment depends upon implementation, methodology, and application.

- Differences are enough to show distinctions from others and contributions.

**Discussion:**

- Conventional cache performance measures should be extended. Taking the medium, or total measurements considering recent time intervals may not be enough to express the distinction, importance, and uniqueness of this work. More work needs to be done to explore the beauty of the idea.

- Distinguishing performance measures for prefetched objects and objects may show differences and improvements more clearly.

- It may not be enough just looking at object hit and byte hit factors. Byte hit factor may explain web performance more explicitly than hit factor.

- Why is object byte hit factor better than object hit factor? It depends upon object size. Size of the object has more value than counting hits.

- A multiple cache replacement policy will achieve better web performance.

## 4.4 Distinction from Other Studies

Web object prediction is better because the instructor and students follow the same curriculum and their web object requests are highly overlapped. The other methods make predictions by using previous object or client request patterns. When near future web requests are not derived from the previous/current client/object request patterns, other studies have no knowledge for the future object demands. The new method has knowledge for the expected future demands by the instructor.

Prefetched objects will have a higher priority for staying in the local cache for a certain time period as explained above. Other predicting techniques have one object replacement policy for regular objects and prefetched objects. When prediction gets worse, some of the objects requested by the clients have to be replaced with prefetched objects. Cache performance for a group should be considered because predicted objects are requested during the particular time interval.

The other prefetching studies predict future object requests while current client requests are processing. Since network bandwidth is limited during the school hours, limited network bandwidth may be wasted because of prefetching and the waiting time of clients will be higher. However, the instructor initiated prefetching technique fills the cache during the light network traffic hours (the network is most probably idle). Thus, network bandwidth utilization will be better than other methods.

This method is better for a group of clients. When those clients follow the instructor well enough, web performance is much better.

## 4.5 Judgment of this part

Assumptions:

1. The instructor guides a group of students.

2. The instructor and students have to follow the curriculum.

3. The instructor should know what to read/see for the students and the curriculum.

**The instructor (he or she):**

1. S/he is responsible for following the curriculum because of her/his position.

2. The instructor has to give accurate guidance to the students.

3. The instructor should ask himself whether his guidance is enough, accurate, appropriate in content, time, etc. to accomplish the mission.

- Instructor should give the appropriate objects to the students.

- Suggested objects should be brought to the local domain to avoid long object travel time. Students may loose interest if some suggested objects which are highly useful are not returned quickly.

4. The instructor makes a decision for a group of students. If he fails to bring objects that are not appropriate, accurate, or useful to students, the mission was not successful because of the instructor. Even though the instructor may claim that those objects may be useful, his approach, accuracy, or timing may not be enough to see satisfactory results.

**Students:**

1. The students will be happy when

- they see something in front of them without waiting

- objects are (observed, understood, consumed, etc.) appropriately by them

- appropriate motivation from the instructor is given on time

2. The students are not happy when

- their interest are not highly overlapped with instructor's interest (the instructor and those students are not in the same domain)

- results of students are not recognized or seen by the instructor

3. Role of the students are:

- follow the instructor, or

- if he guides to wrong direction, the students may/may not find the right direction.

**Mission:**

1. The instructor should guide the students. If he is not eligible, he should not. If the instructor takes the responsibility, he should do his best.

2. The instructor should have enough ability to understand the problem, students, objects, system and expectations from this mission.

3. The instructor should know the resources, their availability, etc.

4. Methodology of the instructor is extremely important.

5. Students should follow the instructor.

As a conclusion, everything should be judged in its domain. Sometimes observed results may not be enough to judge the instructor, students, method and the system. Time will answer.

## 4.6 Contributions

In addition to results of the instructor initiated prefetching, the following results can be seen.

1. It can be used for a group of clients, interest groups, and personal interest to share the same objects via the proxy cache.

2. Prefetching is better because the instructor does it.

3. Prefetched objects are brought while network resources are lightly used.

4. The new cache replacement policy makes sure that prefetched objects stay in the cache for the time interval.

5. This method utilizes resources more effectively by predicting efficiently, bringing to the local cache effectively, and keeping in the cache for a time interval.

6. When all the users follow the instructor, web performance will be much better and efficient.

7. It can be extended to multiple interest groups or proxy servers by building hierarchies.

## 4.7 Future Work

- Latency measurement should be developed.

- Different cache replacement policy under the circumstances should be explored.

- More experimental results with various web performance factors should be collected to be accepted by scientists to prove this work has some value. Some of those factors such as a group of clients, objects, networks, conditions, and time interval for requests, prefetched object aging function, the cache utilization (storage capacity, speed, etc.), proxy server factors (process throughput, request handling, etc.) need to carefully addressed.

- Comparison may be difficult unless the working conditions of methods are explained explicitly.

- Advanced measurement issues for group, client, requests, objects, and working schedule should be distinguished to see a clearer picture.

- Using prefetched objects with other interest groups is a novel approach.

If experimentation is not possible or appropriate, simulation may be done.

## 4.8 Conclusion

We successfully completed the instructor initiated prefetching model to collect experimental results. The instructor submits search criteria online and also has some preferences to choose such as search engines, preferred links, etc. Our experimental results support that the instructor initiated prefetching contributes additional object hit ratio. Eventually client perceived latency would be reduced. This experimental result also supports that those prefetched objects are only needed for a certain time (scheduled working hours with prefetched objects). As soon as that time is passed, those prefetched objects are rarely requested. The new cache replacement algorithm supports that prefetched objects have a higher chance of staying in the cache for a certain time and a lower chance for other times than other objects.

More work needs to be done on how instructor inputs affect cache performance. For example, varying some instructor preferences may affect cache performance. Web performance also depends upon how well students follow the instructor. Since students have a choice on the web, their interest may not overlap with the instructor's very well.

# Chapter 5

# Web Performance

*Don't take responsibility if you cannot handle it*

*A good companion shortens the longest road*

*Define your target, follow your heart*

*Those who talk know how to distribute;*

*those who reserve know how to collect*

*One is to much, many results are too little to show*

*Comparison is projection of things in a domain*

*Everything is judge-able in its domain*

In this chapter, the components of a web request are explored. Then, web performance is defined and compared for several cases. The goal is to reduce the waiting time of clients while utilizing web resources more effectively. One way to reduce object retrieval time is to make web requests while network traffic is lightly loaded. Our approach called **the instructor initiated prefetching** is explained in the previous chapter. This Chapter explores methodology and fundamentals of the web performance. The collected measurements have some meaning if the experiment environment is specified clearly and exactly in addition to expectations. In short, the collected data gave some idea about variation of latency and how difficult it is to collect and comprehend the data.

## 5.1 Introduction to Web Performance

Web performance is from the client's point of view – the smaller the client waiting time, the better web performance. How do we define waiting time of a client? This part addresses some issues in latency (client waiting time, object retrieval time). One approach is the total amount of time to retrieve the whole object. Another approach would be the time to retrieve the first byte of the object. The better approach would be partial object retrieval time (e.g., retrieval time of 10%, 25%, 50%, 75%, or 100% of the object size). This part of the work explores latency definitions, measurements and comparisons for some cases as a web performance.

Web performance from the web server's point of view is that the web requests should be responded to as soon as they come to the web server. A smaller waiting time and faster web request responses at the web server lead to better web performance.

Web performance from the network's point of view is that data should be transferred without any delay or congestion at the maximum available transfer rate in the transmission medium.

Since web performance from the client's point of view considers all web (client, server, and network) resources in this work, web performance means web performance from the client's point of view unless it is otherwise specified. Various waiting times of the clients (latencies) are defined in the following sections.

## 5.2 Components of the Web

To clarify of our discussion, there are a few definitions:

**Object** = name, address (name or IP address) such as a sample object ~/tboult/index.html, www.eecs.lehigh.edu

**Web object** = protocol, object. For example, HTTP, port 80 and the sample object.

**Request web object** = request method, web object. For example:

GET, web object

**Client:** Makes a web request from the proxy or the web server through the Internet.

**Local Area Network (LAN):** Every computer is connected to the LAN. The LAN has some interconnection topology and is connected to the Internet.

**Proxy:** One of the computer usually is assigned to serve the clients in the LAN. Some of the recently accessed web objects stay in the local cache. When another client requests an objects in the cache, it will be retrieved from the proxy. If the requested object is not valid or found in the cache, it is requested from the web or other proxy server on behalf of the client.

**Internet:** The Internet transfers data (request/reply) between two end points.

**Web Server:** It provides the object to satisfy client.

**Cxy:** denotes states at the client

**Pxy:** denotes states at the proxy

**Sxy:** denotes states at the Web server

x is one of (1,2,3,4,5), y is one of (a,b,c).

**Txy:** denotes states transferring data from x to y in the transmission medium. x, y are one of (C, P, S).

A client process (C) has the following states:

C0: check every xc time unit whether there is a client request

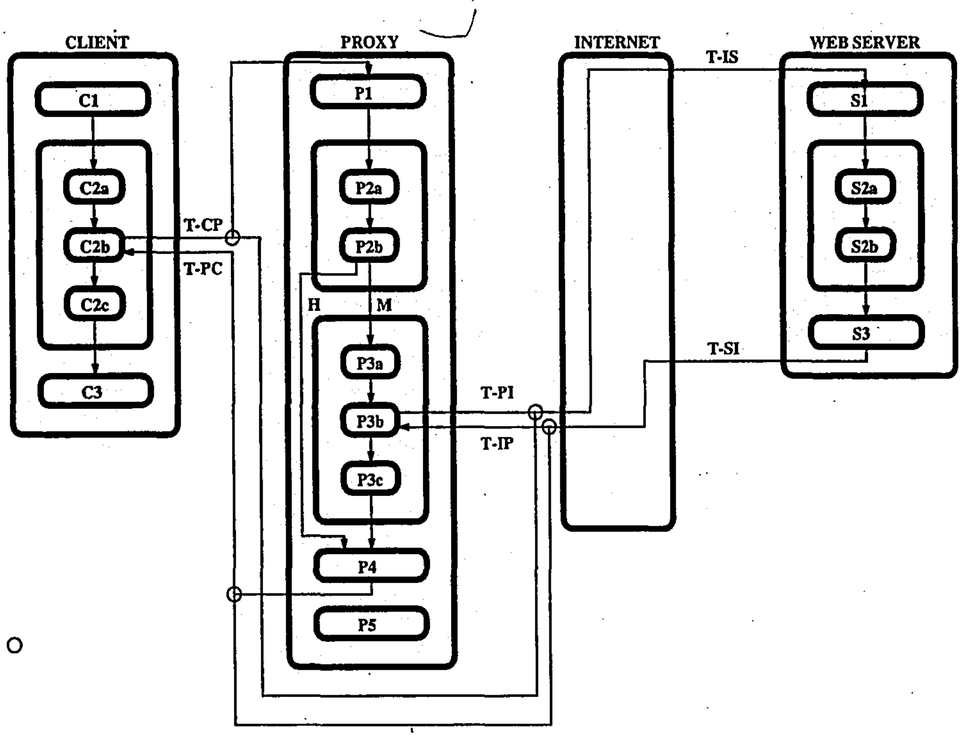

Figure 5.1: Block diagram of the web request states

C1: c-request (client request): read client request input (keyboard, mouse, sound, video, program, script, etc.)

C2: c-wait (client waiting time to retrieve the request) consists of (C2a, C2b, C2c):

C2a: process client request (construct data structure, variable resolution, protocol requirements, etc.) and send client request to the network port of the client computer system

C2b: wait to get reply from the proxy/server

C2c: read reply from the network port of the client computer system and process reply data (combine data packets, reconstruct, encryption data, etc.)

C3: c-reply (client reply): show results (screen, sound, file, execute program, etc.)

State Cx = Collection of states (Cxy) in order where y elements of (a, b, c) and x = 2.

A proxy process (P):

P0: check every xp time unit whether there is a proxy request

P1: pc-request (client request to the proxy): read network port of the proxy server system for the client request and process the request (data, protocol resolution, etc.)

P2: hit-wait (waiting time to search the object in the cache) consists of (P2a, P2b)

P2a: search the requested object in the cache

P2b: If the requested object is found, HIT, read the object from the local storage and go to P4

else MISS, go to P3

P3: miss-wait (waiting time when the object is not found in the cache, e.g., retrieval time of the requested object from the web or other proxy server) consists of (P3a, P3b, P3c)

P3a: process request (construct data structure, variable resolution, protocol requirements, etc.) to send on behalf of the client and send the missed client request to the network port of the proxy server computer system

P3b: wait to get reply/object from the web/proxy server

P3c: read reply from the network port of the proxy server and process received data (combine, reconstruct received data, encryption, etc.)

go to P4

P4: p-reply (waiting time to prepare client reply): prepare proxy-client-reply (construct data structure, variable resolution, protocol requirements, etc.) and send the client request to the network port of the proxy server system

P5: cache maintenance (store requested object, update proxy logs, statistics, client/object info, etc.)

Px = Collection of states (Pxy) in order where y elements of (a, b, c), x = 2, 3

Client service time at the proxy for a client request object (e.g., **Squid** calculates service time between accept() and close() calls):

HIT = pc-request + hit-wait + p-reply = P1 + P2 + P4

MISS = pc-request + hit-wait + miss-wait + p-reply = P1 + P2 + P3 + P4

A web server process (S):

S0: check every xp time unit whether there is a web server process

S1: s-request (client request to the web server): read client/proxy request from the network port of the web server system and process the request (data, protocol resolution, etc.)

S2: s-wait (waiting time to retrieve the request from the server) consists of (S2a, S2b)

S2a: find the requested object

S2b: read the requested object from the local/distributed storage

S3: s-reply (waiting time to prepare server reply): prepare server-reply (construct data structure, variable resolution, protocol requirements, etc.) and send the requested object/reply to the network port of the web server system

Sx = Collection of states (Sxy) in order where y elements of (a, b) and x = 2

Object transmission states in transmission medium:

TCS: client request travels from the network port of the client computer system to the network port of the web server system through the Internet

TCP: client request travels from the network port of the client computer system to the network port of the proxy server system through the Internet (e.g. LAN)

TPS: client request travels from the network port of the proxy server system to the network port of the web server system through the Internet

TSP: server reply travels from the network port of the web server system to the network port of the proxy server system through the Internet

TPC: proxy reply travels from the network port of the proxy server system to the network port of the client computer system through the Internet (e.g. LAN)

TSC: server reply travels from the network port of the web server system to the network port of the client computer system through the Internet

Note: Data transmission of the client request/reply has to follow the states in order. When each client request/reply consists of more than one message (packet stream defines request/reply), those states may not be distinguished one from the other. But each single packet to the client has to follow those states in order. Packets also travel out of order. But before each packet is considered as a client request/reply, those packets need to be combined all or in part to use them.

## 5.3 Web Requests

**Case I.**Web request path via direct connection:

The client request follows the following states in order

C, TCS, S, TSC, C

C1, C2 (TCS, S1, S2, S3, TSC), C3

C1, C2a, C2b (TCS, S1, S2 (S2a, S2b), S3, TSC), C2c, C3

**Case II.**Web request path via web caching:

$$\begin{aligned} objecthit &= C, TCP, P, TPC, C \\ &= C1, C2(TCP, P1, P2, P4, TPC), C3 \\ &= C1, C2a, C2b(TCP, P1, P2a, P2b, P4, TPC), C2c, C3 \end{aligned}$$

$$\begin{aligned} objectmiss &= C, TCP, P, TPS, S, TSP, P, TPC, C \\ &= C1, C2(TCP, P1, P2, P3(TPS, S, TSP), P4, TPC), C3 \\ &= C1, C2a, C2b(TCP, P1, P2a, P2b, P3a, P3b, \\ &\qquad (TPS, S1, S2(S2a, S2b), S3, TSP), P3c, P4, TPC), C2c, C3 \end{aligned}$$

**Case III.**Client initiated (prefetching) web caching

Prefetching occurs while the client request is processing. After the client request resumes, prefetched object requests may still be processing. Thus, the client request and prefetched requests are interlaced. Their request pattern states may not be distinguished. For example:

Since there is no separate proxy for this case, P states occur in the client machine. Object hit and miss states for a client request remain the same, which is given, in case II.

Client request C, and prefetched objects $CP_i$ where i is the number of object requests. Let us consider cases of prefetched object. We will consider best, worst, and other cases.

Prefetched objects:

1.Best prefetch case: Prefetched object requests occur after the client request is completed. Every object request is completed separately. $X_i$ denotes state of the prefetched object I where X is one of (CP, TCS, S, TSC, CP).

```
C, TCS, S, TSC, C
for (i = 1..n) do
  $CP_i$, $TCS_i$, $S_i$, $TSC_i$, $CP_i$
```

2.Worst prefetch case: All the prefetched objects are requested while the current client request is processing.

```
C,
  while (TCS, S, TSC) do
    send $C_i$
$TCS_i$, $S_i$, $TSC_i$
     receive $C_i$,
C
```

C, (TCS, S, TSC, collection of all the $CP_i$ prefetched object request states), C

3.Others: Some of the prefetched objects are retrieved while the current client request is processing. After the client request is completed, some objects are still waiting to be retrieved.

States are between best and worst prefetching cases (cases 1 and 2).

**Case IV**. Server initiated (push) web caching

Client request C, prefetched objects $SP_i$, (i number of prefetch object requests)

C, TCS, S, C and S, ($SP_i$, (TSC, $TSC_i$, $C_I$)), C

1.Best prefetching case: All the prefetched objects are requested after the current client request is initiated. Client waiting time, latency, is not affected by prefetching. Client request states are:

```
C, TCS, S, TSC, C
for (i = 1..n) do
  $CP_i$, $TCS_i$, $S_i$, $TSC_i$, $CP_i$
```

2.Worst prefetching case: The current client request has to wait until all the prefetched objects are requested/retrieved.

```
C, TCS, S
  while (S, TSC, C) do
    send $C_i$
$TCS_i$, $S_i$, $TSC_i$
     receive $C_i$,
C
```

C, TCS, S, (collection of ($SP_i$, $TSC_i$, $C_i$)), TSC, C

3.Others: Some of the prefetched objects are retrieved while the current client request is processing. After the client request is completed, some objects are still waiting to be retrieved.

States are between best and worst prefetching cases (cases 1 and 2).

**Case V**. Proxy initiated web caching

Prefetching can be done while the current client request is processing by using client, object, and server information. But prefetching still introduces extra overhead to the current client request. Similar derivations can be done like case III and IV for best prefetching case, worst prefetching case, and others.

1.Best prefetching case: Client waiting time with proxy does not change. The client request states follow:

C, TCS, S, TSC, C

2.Worst prefetching case: Client waiting time is the total time to retrieve all the prefetched objects and the current client request. This is comprised the following steps:

C, TCP,

(for all objects (client request, prefetched objects) at P:

(TPS, S, TSP),

TPC, C

3. Others: Some prefetched objects are retrieved while the current client request is processing. Some other objects are retrieved after the current client request.

Steps are between best and worst prefetching cases (cases 1 and 2).

**Case VI**. Instructor initiated prefetching

This model is explored in chapter 4 in detail. The following steps need to be completed for each prefetched object.

1.collect instructor specs: C, TCP, P

2.crawl candidate object pointers (URLs): P, TPS, S, TSP, P

3.retrieve prefetched objects: P, TPS, S, TSP, P

Prefetched objects are requested while resources (servers, network connections, etc.) are lightly used.

1.Best prefetching case: All the prefetched objects are brought to the cache while resources are lightly used. Thus, the instructor initiated prefetching neither slows down the current client web request nor consumes current available resources. As a conclusion, the instructor initiated prefetching does not cost anything to the clients during the client request time interval.

There is no worst case nor others like previous prefetching methods. In the literature, we have not seen any discussion of worst cases.

Clients follow the instructor (prefetch object hit):

C, TCP, P, TPC, C

Clients who do not follow the instructor but have similar interests with other clients – after the first access of the object from one of the clients, other client requests for that object are found in the cache (object hit):

C, TCP, P, TPC, C

Clients who do not follow the instructor or have similar interest with the other clients (object miss):

C, TCP, P, TPS, S, TSP, P, TPC, C

If a client does not follow the instructor nor have similar interest with other clients in that class (group), that client always has an object miss. That client needs to:

- Be a leader (instructor even himself) for a group of clients. Thus, they do not need to follow the instructor

- Leave the current group and find another group, which has similar interest with that client

**For all cases:**

Best case: denotes when object hit happens in each case except case I.

Worst case: denotes when object miss happens (e.g., all objects are missed for case I).

## 5.4 Latency: Client Waiting Time

Client waiting time (latency) is a primary issue from the user's point of view. There are two main reasons for latency: processing latency in the end systems and communication latency throughout the network. Processing latency depends upon the load of the end systems. By using a web caching system, the server load will be reduced. The communication latency consists of queuing delay and propagation delay. The more network bandwidth is available, the lower the queuing delay. Increasing network bandwidth may not change propagation delay.

The available network bandwidth between clients and web servers is not used all the time at full capacity. Since most people consume network bandwidth during the day, available network bandwidth is narrow/limited per client. While people are not consuming the network bandwidth (e.g. night, weekend, holiday times), the available network bandwidth is broad sometimes and narrow some other times. A network bandwidth usage should be distributed throughout the time (e.g. day, week, etc.) to distribute the network traffic.

Web request states and various object retrieval methods are given above. Latency definition for a client request would be the sum of the times to complete each state. Then, latency for each case is derived.

Latency for case i by following states = $L_i(states)$,

i = (I, II, III, IV, V, VI)

Latency to complete state j for case i = $L_i(j)$, e.g., j = (nodes (C, P, S), travel (Txy, x and y are nodes)).

$L_{VI}(C2b)$: waiting time of the clients to retrieve the object from either the proxy or web server for instructor initiated prefetching method. The object is returned from the cache in case of object hit. Otherwise, the object is requested from the original web or

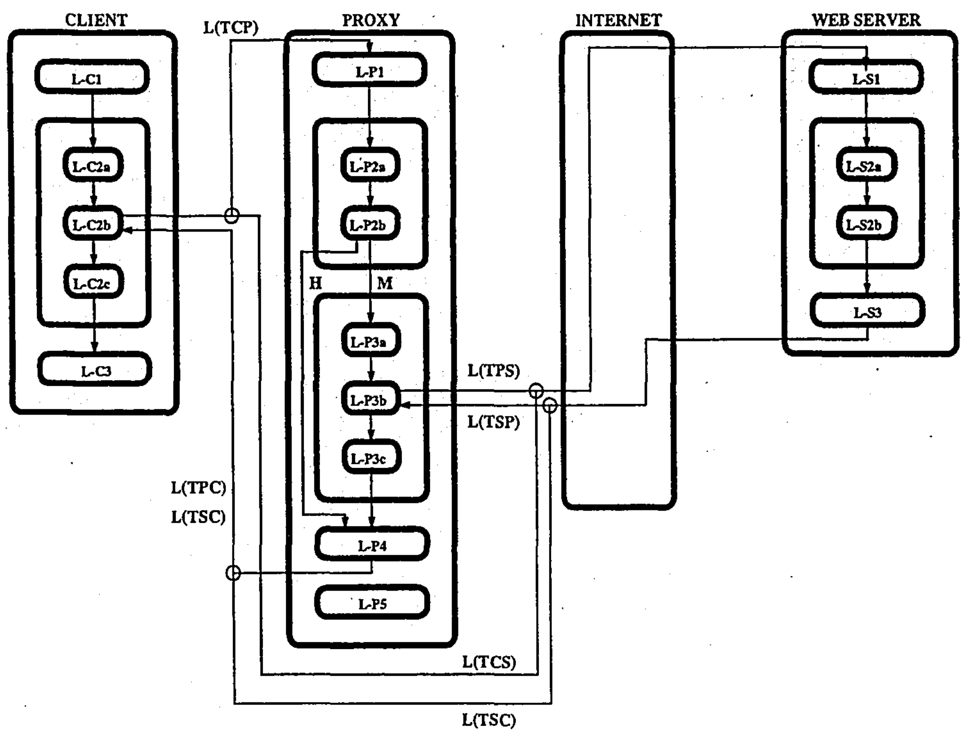

Figure 5.2: Web request latencies for each state

other proxy servers. The request and reply travels through the transmission medium.

$L_{VI}(TCP)$: time to transfer data from the client system to the proxy system

$L_{VI}(hit)$: client waiting time if the requested object is found in the proxy by using the instructor initiated prefetching method

$L_{V.1}(miss)$: client waiting for the best prefetching case if the object is not found in the proxy by using the proxy initiated prefetching method

Let h and m represent probability of the object being either in the cache or not respectively. When object hit and miss probability are unique, latency for case i is:

$L_i$ = h * $L_i(hit)$ + m * $L_i(miss)$

$h$ and $m$ are function of object hit and object miss respectively. $L_i(hit)$ and $L_i(miss)$ are average waiting time for object hit and miss respectively.

### 5.4.1 Latency For an Object

Latency to retrieve an object for each case is derived below. It is assumed that each web request has the same characteristic (e.g., C2a is the amount of time to process and send client object requests to the network port of the client system) and is more or less the same for each case. All object data is transferred from one state to another.

Object = A unit size object O is located at the server S.

Case I: Direct connection. The client request (O) is retrieved from the original web server (S).

$L_I = L_I(C1, C2a, C2b(TCS, S1, S2(S2a, S2b), S3, TSC), C2c, C3)$

Case II: O is retrieved via web caching. The object is either in the cache or not (hit/miss).

$L_{II}$ = h * $L_{II}(hit)$ + m * $L_{II}(miss)$

= h * $L(C1, C2a, C2b(TCP, P1, P2a, P2b, P4, TPC), C2c, C3)$ +

m * $L(C1, C2a, C2b(TCP, P1, P2a, P2b, P3a, P3b, (TPS, S1, S2$

$(S2a, S2b), S3, TSP), P3c, P4, TPC), C2c, C3)$

Case-III: O is retrieved by the client initiated (prefetching) web caching method.

$L_{III.1}$ = O is retrieved with the best prefetching case by this method. O is already prefetched before it is requested. Latency is the local object access time. However, this method retrieves some other objects as soon as current client request is resumed.

$L_{III.1} = L_{III}(C)$,

Additional load to the system is $L_{III}(C, TCS, S, TSC, C{+}CP_k, TCS_k, S_k, TSC_k, C_Pk)$, k denotes k-th prefetched objects.

$L_{III.2}$ consider example, which is the worst prefetching case for this method.

$L_{III.2}$= $L_{III}(C(TCS, S, TSC,$ collection of all the $CP_k$ requests ), $C$

$L_{III.3}$ = some objects are prefetched while the current client request is processing.

After the client request is completed, some other requests are still waiting to be resumed. Thus, the waiting time for clients is between the best and worst prefetching case depending upon how much extra load prefetching causes.

Case IV: O is retrieved by the server initiated (push) web caching method. Similar latency derivation can be done like the previous case.

Case V: the proxy server retrieves O. Similar latency derivations can be derived like previous cases (IV and V).

Case VI: O is retrieved via web caching. The object is found in the cache (prefetch object hit or prefetch) if the user follows the instructor. If the user has similar interests with other group members but no use from the instructor initiated objects, the object can be found in the cache (object hit). Otherwise, the client request always is missed. p denotes probability of prefetch object hit, h and m denote object hit and miss respectively.

$L_{VI}$ = p * $L_{VI}(prefetch)$ + h * $L_{VI}(hit)$ + m * $L_{VI}(miss)$

$L_{VI}(prefetch) = L_{VI}(hit)$, then

$L_{VI}$ = (p + h) * $L_{VI}(hit)$ + m * $L_{VI}(miss)$

= (p + h) * $L(C1, C2a, C2b(TCP, P1, P2a, P2b, P4, TPC), C2c, C3)$ +

m * $L(C1, C2a, C2b(TCP, P1, P2a, P2b, P3a, P3b, (TPS, S1, S2$

$(S2a, S2b), S3, TSP), P3c, P4, TPC), C2c, C3)$

**Comparison of I, II, VI:**

If p, h, m are not equal to zero, then the biggest latency belongs to case I, smallest latency belongs to case VI, and latency for case II is between case I and case VI.

$L_I > L_{II} > L_{VI}$

Thus, the instructor initiated prefetching gives smallest latency.

For example, the client waiting time to request the following objects from the closed-i.eecs.lehigh.edu as a squid user (More information can be found in [see Appendix-2: latency measurement]). The following objects (O1, O2, ..., O7) are retrieved directly or via proxy cache.

| | Case I | Case I | Case I | Case II | Case II | Case II | |
|---|---|---|---|---|---|---|---|
| Object | first | second | third | first | second | third | status |
| O1 | 0.17 | 0.11 | 0.12 | 0.13 | 0.12 | 0.09 | m h h |
| O2 | 0.12 | 0.12 | 0.09 | 0.14 | 0.11 | 0.09 | m h h |
| O3 | 0.31 | 0.11 | 0.10 | 0.34 | 0.13 | 0.08 | m h h |
| O4 | 11.95 | 6.12 | 1.64 | 1.32 | 0.09 | 0.21 | m x x |
| O5 | 1.16 | 0.12 | 0.13 | 0.12 | 0.11 | 0.12 | m h h |
| O6 | 0.70 | 0.39 | 0.41 | 0.69 | 0.48 | 0.44 | m m m |
| O7 | 0.54 | 0.40 | 0.33 | 0.42 | 0.27 | 0.08 | m h h |

Table 5.1: Latency measurement of Lat-2 cases I, II

```
O1: http://closed-i.eecs.lehigh.edu/index.html
O2: http://www.eecs.lehigh.edu/index.html
O3: http://www.eecs.lehigh.edu/~vast/index.html
O4: http://www.lehigh.edu/index.html
O5: http://www.cs.columbia.edu/index.html
O6: http://gohawaii.about.com/cs/volcanoes.index.htm
O7: http://www.geology.sdsu.edu/how_volcanoes_work/Hawaiian.html
```

lat-2: latency collected on Mon Aug 20, 2001 at 10-11AM

lat-4: latency collected on Wed Aug 23, 2001 at 2-3AM

lat-5: latency collected on Wed Aug 23, 2001 at 10-1030AM (cache only)

The following data were collected from unix shell by using **time, wget** utilities.

H is prefetching contribution as an object hit.

h is object hit, m is object miss, and x is object hit and miss (client request requires more than one file).

Summary of service time (msec) (elapsed time between accept() and close() ) for case II in the tables:

**Data Analysis:**

Comparison of Lat-2 and Lat-5:

| | Case I | Case I | Case I | Case II | Case II | Case II | |
|---|---|---|---|---|---|---|---|
| Object | first | second | third | first | second | third | status |
| O1 | 0.10 | 0.11 | 0.13 | 0.10 | 0.11 | 0.10 | m h h |
| O2 | 0.14 | 0.15 | 0.11 | 0.15 | 0.14 | 0.10 | m h h |
| O3 | 0.47 | 0.16 | 0.17 | 0.92 | 0.14 | 0.16 | m h h |
| O4 | 2.69 | 0.26 | 1.33 | 1.32 | 0.24 | 0.10 | m x x |
| O5 | 0.17 | 3.48 | 0.13 | 0.15 | 0.11 | 0.12 | m h h |
| O6 | 1.07 | 9.13 | 0.34 | 0.49 | 0.48 | 0.52 | m m m |
| O7 | 0.53 | 0.39 | 0.40 | 0.46 | 0.27 | 0.21 | m h h |

Table 5.2: Latency measurement of Lat-4 cases I, II

| Object | first | second | third | status |
|---|---|---|---|---|
| O1 | 0.16 | 0.11 | 0.13 | H h h * |
| O2 | 0.13 | 0.11 | 0.13 | H h h * |
| O3 | 0.12 | 0.14 | 0.11 | H h h * |
| O4 | 3.11 | 0.24 | 0.09 | m x x |
| O5 | 0.09 | 0.10 | 0.12 | H h h * |
| O6 | 0.65 | 0.55 | 0.50 | m m m |
| O7 | 0.16 | 0.18 | 0.21 | H h h * |

Table 5.3: Latency measurement of Lat-5 case II

| Object | min | mid | max |
|---|---|---|---|
| O1 | 8 | 9 | 34 |
| O2 | 8 | 9 | 35 |
| O3 | 8 | 10 | 230 |
| O4 | 30 | 49 | 1207 |
| O5 | 9 | 25 | 40 |
| O6 | 397 | 435 | 499 |
| O7 | 11 | 101 | 368 |

Table 5.4: Client service time of Lat-p-2

| Object | min | mid | max |
|---|---|---|---|
| O1 | 8 | 9 | 13 |
| O2 | 8 | 10 | 45 |
| O3 | 8 | 9 | 818 |
| O4 | 30 | 48 | 1270 |
| O5 | 9 | 24 | 70 |
| O6 | 374 | 374 | 433 |
| O7 | 11 | 100 | 410 |

Table 5.5: Client service time of Lat-p-4

| Object | min | mid | max |
|---|---|---|---|
| O1 | 7 | 8 | 9 |
| O2 | 7 | 8 | 9 |
| O3 | 8 | 8 | 10 |
| O4 | 27 | 179 | 3055 |
| O5 | 7 | 8 | 9 |
| O6 | 397 | 430 | 433 |
| O7 | 9 | 10 | 11 |

Table 5.6: Client service time of Lat-p-5

Both are collected during the same time interval of the day. The instructor initiated prefetching objects in the local cache supports lat-5. Then, first request time of the object is reduced to object hit time.. The latency for the O2 and O7 are very significant. When the object changes (dynamic object) for each client request (O6), the object has to be retrieved from the original server.

Case-VI: Object hit for case-II will be prefetch hit if the client follows the instructor's guidance.

Why does latency for a particular object varies so much?

For example, O4: Lehigh University's home page.

To get Lehigh Universities home page took 11.95sec (direct) 1.32 sec (object miss via proxy) in Lat-2. When it was retrieved from the local proxy, it took (0.09 - 0.20 sec). When the same object was requested in the following day, it took 3.11 sec (object miss) and 0.24 sec (object miss and hit) in Lat-5. Lehigh University's networks and some servers are very slow, especially at night. Even though, some people expect better service during the night times and some backup systems consume resources, service quality and client waiting time during the night become worse depending upon when, where, what, and how to collect data while implementing this method.

On the contrary, object O5 (Computer Science department of Columbia University) took 1.16 sec (direct) and 0.16 sec (object hit via proxy) on Lat-2. When the same object requested in Lat-5, it took 0.09-0.12 sec to retrieve from the proxy.

It is obvious that there is tremendous amount of variation. The numbers actually collected are probably not indicative of any one of these problems. Therefore, this work does not concentrate on the conclusions and the data we measured. In the general paradigm, you might end up measuring in your environment.

Some of the factors causing long latency: object (size, content), sever name, request time, request method from the server (whether a redirection to another server or place), latency reasons from the server (client request processing time, searching data base, freshness of the object, etc.), client server system (resource utilization), proxy factors (size, speed, the number of request to handle, object search in the data base, transferring data from local storage to the network port of the proxy server system, etc.), network factors (availability of networks, data transfer path, available data transfer rate, quality of the transfer medium, etc.), prefetching time (each server and network component has different utilization or availability capacity throughout a time interval, finding an optimal time is the open problem as far as we know).

Results of this work will help to answer some of the questions – contribution of the instructor initiated prefetching and effective resource utilization (server, network, proxy). Making web requests in advance while resources are used lightly is a novel approach to the web requests and web caching.

**Comparison of case III, IV, V, VI:**

Since prefetching is done while the current client request is processing for cases III, IV, V; comparison of latency for those cases depends upon implementation of prefetching method and object and client request patterns. Thus, comparing numerical results for a certain experiment is not enough nor appropriate. Prefetched objects are brought in advance by the instructor initiated prefetching. Cases III and IV are summarized in chapter 3. In addition, case V is introduced here. The proxy has more knowledge about clients and objects. Making prefetching decisions have advantages. But, its benefit can be limited when prefetching consumes available web resources while the current client request is processing. On the other hand, retrieving prefetched objects in advance while web resources are lightly used has more advantages than other prefetching methods. A detailed explanation is given in Chapter 4. As a conclusion, the instructor initiated prefetching gives the least latency among others.

**Comments:**

- Since each client request goes to the cache server, the cache server has to look up whether that requested object in the cache or not. This time is called cache processing time $L_{II}(P2a)$. If the object is found in the local cache, it needs to be copied from the local cache storage and transferred to the client (P2b, P4, TPC).

- Probability of object hit (h) is a ratio of object hits to the total request when each one of the objects has equal chance. Taking only the probability or ratio of hit/miss may give some idea of web request natures and handling requests. But, having those ratios and probabilities are not enough to represent web performance. For example, having a high hit rate for small size objects and a low hit rate for large size objects may give inaccurate performance measurements when the average hit ratio is calculated.

- The average or median hit ratio and request service time have been considered as the performance measures by previous studies. However, average or median measures may not give appropriate web performance improvement. Having more information for object hit or service time is more appropriate than that of one single data. For example, 24 hit ratios per day has more information than one hit ratio per day.

- Even if the object hit ratio is increased, the other latency factors may become important (e.g., finding the object in the cache, client request process speed, retrieval time from the local disk, maintanence of the cache, and more).

- Some of the performance measurements by web caching consider web performance as proxy cache performance. Since a proxy cache server is located between clients and the Internet (including servers), denoting cache performance as web performance may not be suitable. Web performance consists of client, the Internet, web server, and cache server factors. The derived latency definition holds for those factors above. Thus, proxy cache performance has limited meaning under the specific conditions. If proxy server factors are more dominant than others, the contribution of other factors to the total web performance can be avoided. Conditions have to be stated clearly and accurately. Limited information may misguide and misjudge.

- When the client makes the next request, and the requested object is found in the cache by prefetching, client waiting time may be less. However, client, server, and proxy initiated prefetching methods still introduce extra load to the system. Therefore, it is not easy to get the latency of the requested object.

- If the client request is already in the cache, the client will have the advantage of less waiting time, but first access time takes much longer (including prefetched retrieval time in part). In addition, prefetching consumes more web resources. Some of prefetched requests may waste network resources.

- Objects need to be searched, including prefetched objects in the proxy cache, by prefetching methods. Proxy processing time will be increased. The more web requests on the request path, the less available resources and more latency under the same conditions.

- Prefetching method and implementation are very important in achieving higher web performance.

- A particular web server may have received more than one request to accomplish the current client request.

- The cache server performance becomes important. The cache server has to deal with more requests and maintain more than one object to accomplish a single client request. Thus, the server response time may be longer.

- Some of the web servers may be slower than others. Thus, the same size of objects that are equal distance from the client may give various latency measurements. An object distance from clients can be defined as the total number of hubs required to reach the object. Thus, the object distance from the clients is not enough to represent web performance alone.

- An object request for a variety of times may have very different latency measures. Some of the reasons are dynamics of Internet traffic, server loads, or even client request system.

**Why the Instructor Initiated Prefetching?**

A group of objects which are highly suggested by the instructor are brought to the proxy server while web resources are lightly used. When a group of users work on the web under the guidance of the instructor and those group members truly follow the instructor guidance, all the client requests will be found in the local cache. The client waiting will be smaller than direct connection from the web or proxy server(s). When the clients ignore the instructor's guidance, they have to wait to retrieve objects. However, some clients may have similar interests. Then, first client requests are the dominant latency factor to retrieve the object from the web server. The following requests for the same object will be returned from the local cache.

Contribution of the instructor initiated prefetching will be explained later. In short, our approach will resolve many problems for the web requests as stated throughout this work.

Since instructor initiated prefetched objects stay in the cache only for a time interval, cache maintanence and cache server performance are better.

The performance improvement (e.g. object hit probability) depends upon how well clients follow the leader (instructor).

Since the instructor makes requests while the network is lightly loaded (or idle in part), prefetching does not consume web resources during the client request processing time duration. Available network bandwidth is constant during the experiment. Consuming some to get benefit will improve web performance.

Making web requests during the light network times distributes client requests instead of high web request demand during the daytime and low elsewhere. Conventional web requests are processed and objects are retrieved as soon as the web request is made. This method makes predicted web requests in advance. The instructor makes web requests in advance and request those objects via the proxy cache for a particular time interval. Making web requests in advance while utilizing web resources more effectively will produce more distributed web resources and better web performance. It will change the nature of the web scheme. Finally, traditional performance measures may not be suitable. New performance measures need to be explored.

As a conclusion, prefetching objects for a group of users working under the guidance of the instructor while web resources are lightly used will reduce client waiting time. The instructor has the responsibility of deciding what to prefetch and the clients have responsibility to follow the instructors guidance. When interests of the instructor and the clients are different and clients have no common interest for a time interval, the prefetching and proxy web caching are not appropriate solution for this case. In reality, some objects are requested by some clients during the particular time interval.

**Comments:**

- Probability of each web request is not the same. Some web objects may have higher request probability for a certain time interval than others. Request probability and object hit probability may be constant under certain conditions.

- Looking at object hit probability or object hit ratio may not give enough web performance data. It is useful at some point.

- Most of the studies determine object hit ratio for a long time duration (several weeks or months). Looking at object hit ratio for a certain time interval may have some value for determining satisfaction of a particular interest group over that time interval.

### 5.4.2 Average and Median Latency

When latency for each request varies, previous work needs to be extended.

**Average:**

For $n$ requests, let $l_i$ denotes latency of each web request $i$. The average latency would be:

$l = 1/n$ (Sum of all request latencies)

$l_i = l_h$ for an object hit, otherwise $l_m$.

For example, latencies are 27, 179, 3055 for object O4 in Lat-5 [see app-latency]. 27 is object hit, 3055 is object miss. 179 is also object miss. Taking the average (27 + 179 + 3055)/3 = 1087. When most of the requests are an object hit, then the average will be low. One big latency may have an effect on many small latencies. Standard variation is very large. More data should be collected for this case to be sure about collected results. We need to measure and collect a large amount of data to make a conclusion by experimental results that this method really works under the specified experimental setup. In short, methodology of this work improves web performance as long as implementations and collected results are designed/analyzed carefully.

**Median:**

Median latency is calculated by using previous requests.

After sorting all latencies, the number in the middle will give median latency. Again,

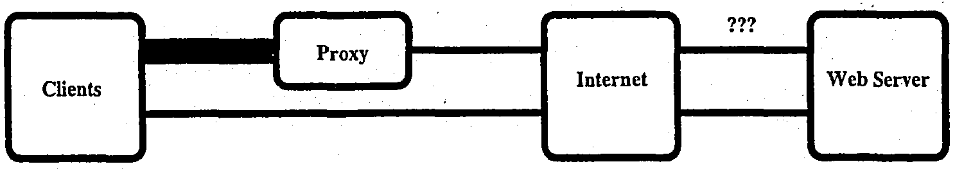

Figure 5.3: Available network bandwidth either direct or proxy connection

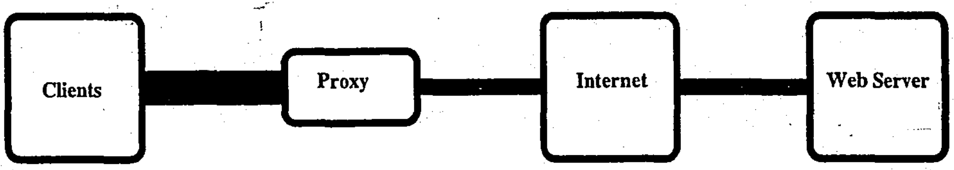

Figure 5.4: Available bandwidth at light loading and proxy fill duration by the instructor initiated prefetching

One single number is not enough to explain latency. For example, latencies are 27, 179, 3055 for object O4 in the Lat-5 [see Appendix-2: latency measurement]. 27 is object hit, 3055 is object miss. 179 is also object miss. Taking the median solely is not enough.

**Comments**

- When client waiting time for each request is not distributed well, such as some big latencies and some small latencies:

1.Taking the average of those will be somewhere in the middle. It is rarely possible.

2. Having median will lead to the same result.

- Collected latency information should be considered carefully.

## 5.5 Variable Object Size and Available Network Bandwidth

Available network bandwidth can be defined by one single number for a time interval. See object hit vs day/hours in Chapter 2.

### 5.5.1 Latency for Varying Object Size and Web Resources

When available bandwidth between clients and web servers varies, client waiting time varies even though the object size remains the same. In general, object size and web resources vary.

Let $s$ be size of the object,

Latency from above,

$l = h * L(hit) + m * L(miss)$

For example:

During the Lat-5 experiment, data transfer rates were collected to retrieve `http://www.lehigh.edu/index.html` from `closed-i,eecs.lehigh.edu` by **wget** utility.

First: 17.73Kb/s (object miss) www2.lehigh.edu — closed-i Second: 14.4 Mb/s (object miss, hit) Third: 9.43 Mb/s (object miss, hit)

Although EECS department is in the Lehigh University's LAN, available data transfer rate varies a lot (ratio is more than 800), other factors play an important role as well.

This small example shows that taking one parameter to represent latency is not enough to show performance. As stated in this work, latency consists of client, network, and servers factors. Eliminating one of them in a certain situation may mislead the judgment. After defining working conditions explicitly and carefully, some of the latency factor's contribution may not be significant (negligible).

The network bandwidth between two ends is defined as the number of transferred bits for an unit time. Its unit is bits/seconds. Available bandwidth between clients and the cache server is $b$, available bandwidth between the cache server and web servers are $b_1$ and $b_2$ at low and high demand times. When the network capacity between the cache server and the Internet is constant, available network bandwidth varies depending upon how many clients are currently using the network. Let us say the maximum available network bandwidth is $b_1$ if no one is consuming the network resources. When $m$ number of clients in the same local network are using the resources, available bandwidth for each one is $b_1/m$ if their network consumption is evenly distributed. Waiting time to send an object is the size of the object divided by the available network bandwidth.

Waiting time if the object with size $s$ is requested from the cache server, $s/b$.

Waiting time if the object with size $s$ requested from the original server or other server, $s/b_1$ and $s/b_2$ respectively low and high network demand times.

Let $L(Txy)$ denotes transferring an object from point x and y.

$L(Txy) = s/b$

When available bandwidth varies between $b_1$ (high network capacity, low demand) and $b_2$ (low network capacity, high demand), $L(T_{xy})$ varies from $s/b_1$ to $s/b_2$.

When the cache server is employed in the same local area network with clients, available bandwidth between clients and the cache server is much larger than that of the Internet.

**Comparison of network connections**

TCS: connection from clients to servers.

TCP: connection from clients to proxy

TPS: connection from proxy to servers

TCS and TPS are similar when clients and proxy are in the same LAN. Thus,

$L(TCS)$ or $L(TPS) > L(TCP)$

TSC: connection from servers to clients

TSP: connection from server to proxy

TPC: connection from proxy to clients

TCS and TSC, TCP and TPC, TPS and TSP may not be the same path to transfer data through the Internet. The sending direction of data varies. For example, download bandwidth is larger than upload bandwidth for some services (wireless, DSL connection, etc.).

Bringing objects from the proxy server takes less time than that of original servers via the Internet. The novel contribution of this work is retrieving objects while resources are lightly used (service utilization is low, availability is high) and making a copy of the objects in the local storage. To be able to use high resource capacity, clients requests need to be given in advance. The intelligent model should also know availability of the resources.

**Latency Derivation of Variable Network Bandwidth for the Instructor Initiated Web Requests:**

$b$ represents data transmission bandwidth between clients and the cache server.

$b_1$ represents data transmission bandwidth during the low latency times (e.g. night, weekend, holidays, etc.). It has maximum available bandwidth when nobody consumes.

$b_2$ represents data transmission bandwidth during high latency times (e.g, daytime, weekday, regular business times (hours, days, et.c)). When available bandwidth is shared evenly among $u$ users equally, then $b_2 = b_1/u$.

Latency for size $s$ object when the request is placed in either night or day:

$l_{low} = s/b_1$

$l_{high} = s/b_2 = u * s/b_1$

Since available network bandwidth has to be shared during the daytime, daytime latency is $u$ times larger than nighttime latency for the same size object.

Txy_high: transmission latency between x and y during the low available network bandwidth times.

Txy_low: transmission latency between x and y during the high available network bandwidth times.

x and y can be Client, Proxy, Server: (C, P, S)

When Txy_low is the minimum latency while nobody consumes, Txy_high would be $Txy\_low * u$ if $u$ users equally share the network resources.

Assumption and usually the case for latency of the transmission medium

$L(TCS) > L(TCP), L(TSC) > L(TPC)\ L(TPS) <= L(TCS)$

Case I: Latency for the direct connection, object retrieved from the original server

$l = L(miss) = L(TCS, TSC)$

Case II: When a cache server is employed and then latency of the transmission medium is the dominant factor, then:

$l = h * L(hit) + m * L(miss) = h * L(TCP, TPC) + m * L(TCP, TPS, TSP, TPC)$

Case VI: When the instructor initiated prefetching is employed,

$l = (p + h) * L(hit) + m * L(miss)$

$l = (p + h) * L(TCP, TPC) + m * L(TCP, TPS, TSP, TPC)$

$l.best = L(hit)$ if all the client requests are returned from the proxy

$l.worst = h * L(hit) + m * L(miss)$

Those cases are explained previously. Similar derivations for other cases (III, IV, V) can be derived.

When $u$ users have to share the network resources,

$l = u * L$

**Comparison of latency among direct, proxy, prefetching cases**

When h, m, p are not zero,

$L_I > L_{II} > L_{VI}$

The instructor initiated prefetching gives the smallest latency. When the network resources are shared by other users, the latency order for those cases still remains.

When object size and available network bandwidth varies, latency would be

$l.best = l_{low} = s/b_1$

$l.worst = l_{high} = s/b_2 = u * s/b_1$

Case I:

$l = L(miss) = L(TCS, TSC) = u * (s/b) * L(TCS, TSC)$

When clients make requests at high network bandwidth times (low latency)

$l.best = u * (s/b_2) * L(TCS, TSC)$

When clients make requests during low network bandwidth times (high latency),

$l.worst = u * (s/b_1) * L(TCS, TSC)$

Similar derivations for case II:

$l.best = u * (s/b_2) * (h * L(hit) + m * L(miss))$

$l.best = u * (s/b_2) * (h * L(TCP, TPC) + m * L(TCP, TPS, TSP, TPC))$

$l.worst = u * (s/b_1) * (h * L(hit) + m * L(miss))$

$l.worst = u * (s/b_1) * (h * L(TCP, TPC) + m * L(TCP, TPS, TSP, TPC))$

Similar derivations for case VI:

$l.best = u * (s/b_2) * ((p + h) * L(hit) + m * L(miss))$

$l.best = u * (s/b_2) * ((p + h) * L(TCP, TPC) + m * L(TCP, TPS, TSP, TPC))$

$l.worst = u * (s/b_1) * ((p + h) * L(hit) + m * L(miss))$

$l.worst = u * (s/b_1) * ((p + h) * L(TCP, TPC) + m * L(TCP, TPS, TSP, TPC))$

Similar derivation for other cases (III, IV, V) can be done.

**Comparison of latency among direct, proxy, and the instructor initiated prefetching cases**

When h, m, and p are not zero, then

$L_{I.best} > L_{II.best} > L_{VI.best}$ and

$L_{I.worst} > L_{II.worst} > L_{VI.worst}$

The instructor initiated prefetching gives smallest latency. When the network resources are shared by other users, and available network resources varies the latency order for those cases still remains.

## 5.5.2 Hit Probability or Ratio

Object hit probability for any object is assumed the same. In reality it may not be. Some objects may have a higher probability of being requested than others. For example, a group of students who are working under the guidance of the instructor have a scheduled web activity. Their object requests for a time interval can be known if they follow the instructor (and curriculum). For that time interval, some of the objects have more of a chance of being requested than others. Thus, not every object in each time interval has an equal probability of being requested. When they send their object requests through the proxy, it can been seen on the proxy logs. Object miss probability is (1 - object hit probability). Object prefetch probability is a kind of hit probability that the object is brought to the proxy cache for a certain time (prefetching time interval).

When each one the probabilities for each object is the same, then object hit, miss, and prefetch rate can be defined.

object hit rate: the ratio of the total number of object hits to the total number of requests.

object miss rate: the ratio of the total number of object misses to the total number of requests.

prefetch rate: the ratio of the total number of object hits especially brought in advance (prefetched objects) to the total number of requests.

Case I: All requests are miss. $h = 0, m = 1$.

Case II: Some object can be found in the proxy cache. Thus, m and h are not either 0 or 1.

Case VI: Some objects can be found in the proxy (either requested before or brought to the proxy in advance (prefetched times)). Thus, m, h, and p are not zero.

For example: 100 object requests are made. 50 of the them are requested again. 40 of the those second requested objects are brought to the cache in advance.

Case I: Total 150 objects are requested. All requests are retrieved directly from the server.

$m = 1, h = 0$

Case II:

| ratio of | prefetch | hit | miss |
|---|---|---|---|
| case I | N/A | 0 | 1 |
| case II | N/A | 0.3333 | 0.6666 |
| case VI | 0.2666 | 0.3333 | 0.4000 |

Table 5.7: Prefetch, hit, or miss rates of cases I, II, VI

| Latency | time unit |
|---|---|
| case I | 1500 |
| case II | 1050 |
| case VI | 690 |

Table 5.8: Latency of cases I, II, VI

150 objects objects are requested via proxy server. 50 objects are retrieved from the proxy server.

h = 50 / (100 + 50) = 50/150 = 1/3 = 0.3333

m = 1- h = 0.6666

Case VI:

150 objects are requested. 40 objects are brought to the proxy cache in advance.

p = 40 / ( 100 + 50) = 40 / 150 = 0.2666

h = 50 / (100 + 50) = 0.3333

m = 1 - ( p + h) = 0.4000

When each direct request takes 10 time units and each hit or prefetch hit takes 1 time, unit the total latency would be,

Latency for cases I, II, VI:

$L_I = 150 * 10 = 1500$

$L_{II} = 50 * 1 + 100 * 10 = 1050$

$L_{VI} = 40 * 1 + 50 * 1 + (150 - (40 + 50)) * 10 = 40 + 50 + 600 = 690$

**Comparison of hit vs. latency:**

As a conclusion, case VI (the instructor initiated prefetching has the smallest latency.

Thus, looking at only object prefetch, hit, or miss rates are not enough. The web performance from the user's point of view needs to include the web requests, web, and methodology.

Contribution of prefetch can be defined as a factor for case VI:

prefetch hit factor: the ratio of the total number of prefetched hit objects to the total hits (objects found in the proxy cache)

prefetch hit factor = 40 / (40 + 50) = 4 / 9 = 0.4444

What does it mean?

150 objects are requested. 40 of the objects are retrieved from the proxy instead of retrieving from the original server. This reduces first access miss. 50 objects already in the proxy cache are requested again. (150 - 40 -50 = 60) objects are missed and had to be retrieved from the original server. When all the missed objects are found in the cache, can you imagine the latency reduction?

$L(Allobjectsarereturnedfromthecache) =$

$L(bestprefetching) = 150 * 1 = 150$

Now, latency comparison:

$bestprefetching < prefetching < requestviaproxy < direct$

$150 < 690 < 1050 < 1500$

150 / 1500 = the ratio of object hit request latency to object miss request latency, 1/10.

In addition, comparing ratios may not reflect actual improvement. Clients have to wait 1500 time units for a direct request from the server, 1050 time units via proxy server, and 690 time units via prefetching. If all the clients follow the instructor's guidance, the total amount of time to retrieve all the objects would be 150. The reduction is really amazing. 1500 (25 minutes) seconds or 150 (2.5 minutes) seconds.

As a conclusion, solely hit, miss, and prefetch rates are not enough to express web performance. More detailed information is needed. Prefetch factors are very important in determining the contribution of prefetching.

**Generalization of this example**

The number of prefetch, hit, or miss objects are pc, hc, and mc respectively, then

The total number of requests = tc = pc + hc + mc

hit = hc / tc

miss = mc / tc

prefetch hit factor = pc / ( pc + hc)

**Additions**

The latency for each request is considered the same. In the real applications, each object requests takes a different amount of time to retrieve. Thus, object size and available network bandwidth is important.

prefetch size factor = the total size of the prefetched objects / (the total size of the objects found in the proxy cache)

hs: total size of the object hits

ms: total size of the object misses

ps: total size of the prefetched object hit

Case II:

object size (byte) hit ratio = hs / ( hs + ms )

Case VI:

prefetch size (byte) hit factor = ps / (ps + hs)

The implementation of **The Instructor Initiated Prefetching** and collected results can be found in the chapter 4.

## 5.5.3 Object Type

The type of the object has been considered the same for each object so far. In reality, different types of objects are available on the Web. Text, html, binary, audio, image, and video objects are commonly used. In addition, files can be sent as is or encrypted. If the object needs a special treatment at both the server and client ends, this adds extra delay to the total client waiting time.

### 5.5.4 State of the Object

It is assumed that each web request state has to be completed before going to another web request state. Usually, connecting to another server takes quite an amount of time. It is called first connection time. After establishing a connection, data transfer starts. Instead of opening several connections, making multiple client requests after the connection is established has advantages over individual client requests. HTTP/1.1 supports persistent connections. In that case, each individual web request has to follow the same states to complete the client request. But it may be difficult to distinguish one from another. When client request transfers are overlapped, taking individual measurements are not appropriate. In that case, multiple client request patterns need to be considered.

### 5.5.5 Data Transfer Through the Internet

Each request or reply is divided into small pieces called data packets. Those packets travel from one end to another. Once a communication between two end points is established, data segments of the object start traveling through the Internet. Not all the packets have to travel in order nor take the same path. The order and path may vary. Some packets may even be lost on their way to reach the destination.

Dividing and combining packets at each end and handling packet loss may effect total latency. Even though a broad communication medium is established between two ends, those issues may add extra delays to transfer time. Many research studies have been pursuing these problems.

### 5.5.6 Handling Objects at the Client Side

It has been assumed so far that each state has to be completed before going to another one. The total data of the request or reply is transferred from one state to another. Data transfer is not interlaced. In another words, each data movement between each state is independent from one another.

The size of client request data (less than 1Kbytes) is usually smaller than that of client request objects (vary from a few bytes to Mbytes).

Our assumption is appropriate for sending client requests. But, if the requested object is big enough at the client side to show the client, the client server shows some portion of the received object. Requesting big image files, while the object is transferring, the client can seen some part on the screen. Latency measurement is not affected from this. The latency is defined as all of the object bytes need to be transferred to calculate latency. In this case, client waiting time and showing results to the client states are overlapped.

The client is happy since some part of data is shown on the screen. In the beginning of this work, web performance from the client's point of view is defined as a latency (object retrieval or client waiting time). The philosophical approach should be web performance from the client's point of view is client satisfaction. Even though definition is simple, modeling, implementation, measurement, and comparison are difficult.

In the following section a novel latency measurement approach will be given briefly.

## 5.6 Future Work

| size | 1K | 10K | 50K | 100K | all bytes |
|---|---|---|---|---|---|
| 1M | 1K | 10K | 50K | 100K | 1M |
| 20K | 1K | 10K | N/A | N/A | 20K |
| 500 | N/A | N/A | N/A | N/A | 500 |

Table 5.9: Latency for each portion of various object size:

| size | 10% | 25% | 50% | 75% | 100% |
|---|---|---|---|---|---|
| 1M | 10K | 250K | 500K | 750K | 1M |
| 20K | 2K | 5K | 10K | 15K | 20K |
| 500 | 50 | 125 | 250 | 375 | 500 |

Table 5.10: Latency for each portion of various object size:

Client waiting time, latency, has been considered for each request to complete all of the bytes to transfer. Another approach to determine latency would be the latency of some part of the object such as 10, 20, ..90, 100 % of total object size or 1 Kbyte, 10 Kbyte, 100 Kbyte of object size. Instead of dividing by multiple of 10, dividing by 5 has more meaning and requires less data to deal with (record, compute, store, etc.). The following latencies denote from that server through the taken path by that client located at particular place and time.

–: smallest portion of object latency denotes connection establishment (e.g., 10

-: less than half portion of object latency denotes small object size latency (e.g., 25

0: half of the object retrieval time denotes average latency (e.g., 50

+: largest portion of object latency denotes large object size latency (e.g., 75

++: Total amount of latency to retrieve the object.

++ can be any size.

For example:

Since object size varies, taking a constant amount of data bytes to retrieve at the client side is not enough to represent progressive latency.

In addition, retrieval time of some portion of the object may have some value.

Taking one or another measure have limited information. Both of them (latency of some portion of object in size and percentile) have more meaning and information. It will characterize the object retrieval. Implementation of this novel latency measurement approach will require careful design, more time and data analysis tools. Once they are completed, latency of a client request object has 10 variables. It requires 10 times more data storage compared to the conventional methods. Its advantages and meaning will be valuable and important. Time is the answer.

### 5.6.1 Statistic Collection

Current implementation of data analysis collects all the client and object request information. When they need to give requested measurements, they usually consider all the data taken since starting or recent time unit. Squid proxy server computes cache utilization for the last 5 minutes, 15 minutes, 60 minutes, 8 hours, 1 day, 3 days, and all (since started). It only gives detailed information for cache utilization. All other web performance measures are limited (last 5 minutes or 60 minutes). The first one has too much information, the second one does not have enough information.

Measuring web performance for a time interval is important. It shows how effective and efficient the applied method is during the time interval. It can be extended for a group of users, objects, servers, etc.

In addition, measuring performance of object type, network type (Squid cache manager utility program gives proxy usage for each individual IP), etc. gives more understanding for web performance and evaluation methods.

### 5.6.2 Other Web Performance Metrics

Server load, capacity

- Availability to serve objects is important because the source of the object is the web server. Handling requests, finding objects, and replying with the object define server performance.

- When the server sends the client request to another server, this adds extra delay and request handling overhead.

- Interpreting client request and deciding the object is important.

- Searching the requested object in the database brings another extra overhead.

- Moving data from local storage (or another server/data storage system) brings additional delay.

## 5.7 Conclusion

This chapter is mostly about the methodology of evaluating web performance. Small set up experimental data have enormous variation, one needs to collect data on the working environment.

Object hit ratio may not be good enough to represent web performance. Latency (as a web performance measure) comparisons of various web caching and prefetching methods were discussed. Then, latency factors (object size, available network bandwidth, probability of being in the cache, proxy, server, network related issues) were discussed.

If the cache server is employed, latency is a function of cache processing time to decide hit or miss, local cache speed, available network bandwidth between clients and the cache server, network bandwidth between the cache server and the web server, web server load, first access time, cache size and replacement policy, etc.

If prefetching is used in the cache server, latency consists of effectiveness of prefetching plus additional proxy overhead.

If Instructor initiated prefetching is employed, latency reflects the effectiveness of prefetching for a group of users and a time interval. The cost of the prefetched objects to the cache (duration of time staying in the cache, replacement method), retrieval cost of the prefetched object depending on when it is done.

In conclusion, comparison is a projection of things in a domain. To be able to make comparisons, each one of the entities has to be observed in that well defined domain. Everything is judge-able in its own unique domain.

# Chapter 6

# Conclusion

*The outcomes of the systems always have similarities and differences.*
*Understanding those similarities and differences to resolve some problems by implementing brings some others.*
*It is easy to say "Come", difficult to say "Go".*
*Do what your teacher says but not what he does.*
*No matter where you go, your destiny follows you.*
*Work as if you were to live forever; live as if you were to die tomorrow.*

## 6.1 Contributions From This Work

New Prefetching has the opportunity for achieving better web performance:

1. Prediction is better than other methods because the leader or instructor has knowledge for the near future requests. The leader has a large and influence on the clients. It is assumed that most (theoretically all) of the clients will follow the leader's suggestion. Prediction accuracy depends upon how well clients follow the leader.

2. Prefetched objects are brought to the local cache any time before needed ($t_1$); it

can be done right after the request is placed and before the scheduled time ($t_1$) begins whenever web resources are available.

3. Prefetching can be done while utilizing web resources more effectively. This will result in a fast response time from the server (no waiting time in the server process queue) and broad network bandwidth (minimum number of hops with the maximum available bandwidth) times. Therefore, limited web resources are not consumed by prefetching.

4. Prefetching can be done during light web traffic times (light web server and data transfer times = less waiting time in the server process queue and broad network bandwidth). Thus, the waiting time for clients will be much less than that of on demand web requests or even web caching with prefetching. This new prefetching technique does not have any cost to clients.

5. This approach does not waste web resource like other prefetching techniques. Most of the prefetching methods send predicted objects while current client requests are processing. Their prediction is based on previous client access profile and object popularity. When client requests are moved randomly (usually the case), accuracy of prediction may be questionable.

6. Since prediction is much better than others, the majority of client requests are returned from the local cache. Therefore, the waiting time for clients will be decreased.

7. Since objects are brought in advance, unavailable data or busy server problems are eliminated. The system has the latest version of the objects in the local cache.

8. As long as prefetched objects are fresh (valid) between $t_1$ and $t_2$, and returned from the local cache, web performance will be much better than other prefetching techniques. Some of the objects may not be valid or stale during the $t_1 - t_2$ time interval even though they are prefetched. Those objects need to be retrieved again or some intelligence needs to be added to the prefetching mechanism. However, the local cache always has the latest version of the objects.

9. Assuming there is enough local storage capacity or mechanism to hold prefetched

objects in the local cache. If there is, prefetching cost of our approach is nothing but keeping the objects in the local cache. Prefetching is done while web resources are waiting in idle state. Sharing storage capacity with others who have the same interest will reduce the cost.

10.This new method saves network traffic especially if a group of clients has a limited network bandwidth and have to share among others. It is also beneficial for the users who have scheduled web requests for the future. They will get benefit of having objects in advance.

11.Dominant latency depends upon object retrieval time from the local cache and request processing time at the local cache. When the local cache is close to the clients and has broader network bandwidth than that of the Internet, the waiting time for clients will be insignificant.

In addition, hierarchical web caching can be used to benefit many schools in the same school district or country. When each school has the same curriculum to follow, they may use the same prefetched objects. Cache hierarchy becomes a role. Building a cache hierarchy that utilizes available network bandwidth more effectively and shares common objects among the same interest groups will be a new avenue for achieving better web performance (latency).

This cache hierarchy may have some benefits:

1. It will reduce local disk storage for each interest group or clients who share the same objects.

2. It will consume web resources more effectively such as bringing only one copy of prefetched objects for all interest groups in each level. It will reduce web resource consumption and latency indirectly.

3. It will introduce extra processing time to find prefetched objects in the hierarchy. But, some of the previous research explained in chapter 3 has some solution for reducing this processing time.

4. Building interest groups and optimization of web resource usage are well known problems. This problem is not easy to solve especially some of the parameters are random (defined by probability density function such as prefetching accuracy, client requests, network availability, etc.).

5. Building interest groups on top of the cache hierarchy may help to achieve better web performance.

## 6.2 Conclusion

Web caching is one of the methods used to get better performance on the World Wide Web by sharing web objects among clients. Client initiated web caching (prefetching) is a recent technique to reduce client waiting time. The instructor initiated web caching is introduced for coordination of client request patterns, web servers, and network resources. To be able to get higher web throughput, making an interest group is an important research area. Defining a group and its relations among members and other groups and characteristics are important contributions from this research. Contributions of this proposed research are discussed briefly above. In summary, communication among the computers will help to coordinate and cooperate information dissemination in the Internet.

# Bibliography

[ARA+95] Marc Abrams, Charles R.Standridge, Ghaleb Abdulla, Stephan Williams, and Edward A. Fox. Caching proxies: Limitations and potentials. In *4th Int. World Wide Web Conference, Boston, MA*, pages 119–133, December 1995. www.w3j.com/1/abrams.155/paper/155.html.

[Bes96] Azer Bestavroz. Speculative data dissemination and service to reduce server load, network traffic and service time in distributed information systems. In *ICDE'96: The International Conference on Data Engineering, New Orleans, Louisiana*, March 1996. cs-www.bu.edu/faculty/best/res/papers/icde96.ps.

[BLea94] T. Berners-Lee and et al. Uniform resource locater (url). Technical report, CERN, December 1994. http://as.internic.net/rfc/rfc1738.txt.

[BLFF96] T. Bernes-Lee, R. Fileding, and H. Frystyk. Hypertext transfer protocol-http/1.0. Technical report, Network Working Group RFC 1945, May 1996. http://ds.internic.net/rfc/rfc1945.txt.

[CAJ+99] Hua Chen, Marc Abrams, Tommy Johnson, Anup Mathur, Ibraz Anwar, and John Stevenson. Wormhole caching with http push method for a satellite-based web content multicast and replication system. In *4th*

*International WWW Caching Workshop*, 1999. www.ircache.net/Cache/Workshop99/Papers/chen-final.ps.gz.

[CK96] Jeremy R. Cooperstock and Steve Kotsopoulos. Why use a fishing line when you have a net? an adaptive multicast data distribution protocol. In *Usenix'96*, 1996. www.ecf.utoronto.ca/afdp/usenix.ps.

[Cor96] Andrew Cormack. Web caching. Technical report, University of Wales, Cardiff, September 1996. www.jisc.ack.uk/acn/caching.html.

[DAP99] John Dilley, Martin Arlitt, and Stephane Perret. Enhancement and validation of squid's cache replacement policy. In *WCW99*, 1999. www.ircache.net/Cache/Workshop99/Papers/dilley-post-final.ps.gz.

[Dav99] Brian D. Davison. A survey of proxy cache evaluation techniques. In *WCW99*, 1999. www.ircache.net/Cache/Workshop99/Papers/davison2-final.ps.gz.

[DCW96] Gihan V. Dias, Graham Cope, and Ravi Wijayaratne. A smart internet caching system. In *INET'96*, 1996. www.isoc.org/inet96/proceedings/a4/a4_3.htm.

[Fea97] R. Fileding and et al. Hypertext transfer protocol-http/1.1. Technical report, Network Working Group RFC 2068, January 1997. http://ds.internic.netrfc/rfc2068.txt.

[Flo98] Cache Flow. High performance web caching white paper. White Paper, 1998. www.cacheflow.com/technology/wp/.

[GS95] James Gwertzman and Margo Seltzer. The case for geographical push-caching. In *Fifth Annual Workshop on Hot Operating Systems*, pages 51–55, May 1995. www.eecs.harvard.edu/~vino/web/hotos.ps.

[Gwe95] James S. Gwertzman. Autonomous replication in wide-area internetworks. Master's thesis, Computer Science, Harvard College, Cambridge, Massahusetts, 1995. www.eecs.harvard.edu/vino/web/push.cache/thesis.html.

[HiCYO99] Hisakazu Hada, Ken ichi Chinen, Suguru Yamaguchi, and Yuji Oie. Behaviour of www proxy servers in low bandwidth conditions. In *WCW99*, 1999. www.ircache.net/Cache/Workshop99/Papers/hada-0ps.gz.

[HMY97] Abdelsalam Heddaya, Sulaiman Mirdad, and David Yates. Diffusion-based caching along routing paths. In *WCW97*, June 1997.

[HWMS98] John H. Hine, Craig E. Wills, Anja Martel, and Joel Sommers. Combining client knowledge and resource dependencies for improved world wide web performance. In *INET98*. Internet Society, July 1998. www.isoc.org/inet98/proceedings/1i/1i_1.htm.

[iCY97] Ken ichi Chinen and Suguru Yamaguchi. An interactive prefetching proxy server for improvement of www latency. In *INET'97*. Internet Society, 1997. www.isoc.org/isoc/inet97/proceedings/A1/A1_3.htm.

[IKY97] Hiroyuki Inoue, Kanchana Kanchanasut, and Suguru Yamaguchi. An adaptive www cache mechanism in the ai3. In *INET'97*. Internet Society, 1997. www.isoc.org/inet97/proceedings/A1/A1_2.htm.

[ISY98] Hiroyuki Inoue, Takeshi Sakamoto, and Suguru Yamaguchi. Webhint: An automatic configuration mechanism for optimizing world wide web cache system utilization. In *INET98*, July 1998. www.isoc.org/inet98/proceedings/1i/1i_3.htm.

[KKO98] Jussi Kangasharju, Young Gap Kwon, and Antonio Ortega. Design and implementation of soft caching proxy. In *WCW98*, 1998. wwwcache.ja.net/events/workshop/23/SoftCaching.html.

[KW00] Balachander Krishnamurthy and Craig E. Wills. Analyzing factors that influence end-to-end web performance. *Computer Networks*, 33(1-6):17–32, June 2000.

[LA94] Ari Luotonen and Kebin Altis. World wide web proxies. In *1st International Conference on WWW, Geneva*, May 1994.

[LOG96] Alejandro Lopez-Ortiz and Daniel M. German. A multicolloborative push-caching http protocol for the www. In *World Wide Web Conference (WWW5)*, 1996. www.cs.unb.ca/ālopez-o/research_papers/Overview.html.

[MC98] Evangelos P. Markatos and Catherine E. Chronaki. A top 10 approach for prefetching the web. In *INET98*, 1998. www.isoc.org/inet98/proceedings/.

[McC94] Rob McCool. The common gateway interface (cgi). Technical report, National Center for Supercomputing Applications (NCSA), 1993-1994. http://hoohoo.ncsa.uiuc.edu/cgi.

[NSS99] Cache flow 100. White Paper, 1999. www.cacheflow.com/technology/wp/.

[PM96] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve world wide web latency. *ACM SIGCOMM Computer Communication Review*, July 1996. daedalus.cs.berkeley.edu/publications/ccr_july96.ps.gz.

[PM98] Themistoklis Palpanas and Alberto Mendelzon. Web prefetching using partial match prediction. 1998.

[Pos80] J. B. Postel. User datagram protocol. Technical report, ISI, August 1980. ftp://rs.internic.net/rfc/rfc768.txt.

[PrSB99] Pablo-rodriguez, Christian Spanner, and Ernst W. Biersack. Web caching architectures: Hierarchical and distributed caching. In *WCW99*, July 1999. www.ircache.net/Cache/Workshop99/Papers/rodriguez-final.ps.gz.

[rfc81a] rfc791. Internet protocol (ip). Technical report, Information Science Institute, University of Southern California, September 1981. ftp://rs.internic.net/rfc/rfc791.txt.

[rfc81b] rfc793. Transmission control protocol (tcp). Technical report, Information Science Institute, University of Southern California, September 1981. ftp://rs.internic.net/rfc/rfc793.txt.

[RRB98] Pablo Rodriguez, Keith W. Ross, and Ernst W. Biersack. Distributing frequently-changing documents in the web: Multicasting or hierarchical caching. In *Computer Networks and ISDN Systems 30*, November 1998.

[squ] http://www.squid-cache.org.

[Ste98] W. Richard Stevens. *Unix Network Programming*, volume 1. Prentice-Hall Inc., Upper Saddle River, NJ 07458, second edition, 1998.

[TDVK98] Renu Tewari, Michael Dahlin, Harrick M. Vin, and Jonathan S. Kay. Beyond hierarchies: Design considerations for distributed caching on the internet. Technical Report TR-98-04, Department of Computer Science, University of Texas at AustinTX 78712-1188, 1998. ftp.cs.utexas.edu/pub/techreports/tr98-03.ps.Z.

[Tou98] Joe Touch. The lsam proxy cache - a multicast distributed virtual cach. In *3rd International WWW Caching Workshop*, June 1998. www.cache.ja.net/events/workshop/14/lsam_arch.ps.

[VdJM98] Ton Verschuren, Andre de Jong, and Ingrid Melve. Web caching meshes: Hit or miss. In *INET98*, July 1998. www.isoc.org/inet98/proceedings/1k/1k_1.htm.

[VR98] Vinod Vallopillil and Keith W. Ross. Cache array routing protocol v1.0 (carp). Technical report, Microsoft and University of Pennsylvania, Februaray 1998. http://info.internet.isi.edu/in-drafts/files/draft-vino-vinod-carp-v1-03.txt.

[WAAF96] Stephen Williams, Marc Abrahams, Charles R. Standridge Ghaleb Abdulla, and Edward A. Fox. Removal policies in network caches for world-wide web documents. In *ACM SICOMM'96 Conference (Stanford University)*, 1996. http://ei.cs.vt.edu/s̃ucceed/96sigcomm/.

[WC97] D. Wessels and K. Claffy. Internet cache protocol (icp), version 2. Technical report, Network Working Group, March 1997. http://ds.internic.net/internet-drafts/draft-wessels-icp-v200.txt.

[WC98] Duane Wessels and Kimberly Claffy. Icp and the squid web cache. *IEEE Journal on Selected Areas in Communications*, 16(3):354–357, April 1998. www.ircache.net/%ewessels/Papers/icp-squid-ps.gz.

[wco] http://shika.aist-nara.ac.jp/products/wcol.

[Wil98] Bert Williams. Transparent web caching solutions. In *WCW98*, 1998. wwwcache.ja.net/events/workshop/33/cachpaper.html.

[WS97] Craig E. Wills and Joel Sommers. Prefetching on the web through merger of client and server profiles. Technical Report WPI-CS-TR-97-1-2, Computer Science Department, Worcester Polytechnic Institute, Worcester, MA 01609, June 1997. ftp.cd.wpi.edu/pub/techreports/pdf/97_1.pdf.

# Vita

*Muammer Akçay* was born in Eskişehir, Turkey. He received his Bachelor of Science and Master of Science degrees from Anadolu University in 1991 and 1993 respectively. His school was restructured under a new name as Osmangazi University and Department of Electrical-Electronics Engineering in 1993.

During the following years, he was employed as a research and teaching assistant in the Department of Electrical-Electronics Engineering at Anadolu University, Eskişehir, Turkey. He entered the Graduate school of Lehigh University, Bethlehem, Pennsylvania. He was employed as a teaching assistant in the Department of Electrical Engineering and Computer Science and as a research assistant in the Vision and Software Technology (VAST) Laboratory at Lehigh University.

END OF
TITLE