ADAPTIVE RESOURCE MANAGEMENT SCHEMES FOR WEB SERVICES

A Dissertation

by

HEUNG KI LEE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Computer Science

ADAPTIVE RESOURCE MANAGEMENT SCHEMES FOR WEB SERVICES

A Dissertation

by

HEUNG KI LEE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Computer Science

ABSTRACT

Adaptive Resource Management Schemes for Web Services. (December 2009)

Heung Ki Lee, B.S., Chungnam University;

M.S., Chungnam University

Chair of Advisory Committee: Dr. Eun Jung Kim

Web cluster systems provide cost-effective solutions when scalable and reliable web services are required. However, as the number of servers in web cluster systems increase, web cluster systems incur long and unpredictable delays to manage servers. This study presents the efficient management schemes for web cluster systems.

First of all, we propose an efficient request distribution scheme in web cluster systems. Distributor-based systems forward user requests to a balanced set of waiting servers in complete transparency to the users. The policy employed in forwarding requests from the frontend distributor to the backend servers plays an important role in the overall system performance. In this study, we present a proactive request distribution (*ProRD*) to provide an intelligent distribution at the distributor.

Second, we propose the heuristic memory management schemes through a web prefetching scheme. For this study, we design a Double Prediction-by-Partial-Match Scheme (*DPS*) that can be adapted to the modern web frameworks. In addition, we present an Adaptive Rate Controller (*ARC*) to determine the prefetch rate depending on the memory status dynamically. For evaluating the prefetch gain in a server node, we implement an Apache module.

Lastly, we design an adaptive web streaming system in wireless networks. The rapid growth of new wireless and mobile devices accessing the internet has contributed to a whole new level of heterogeneity in web streaming systems. Particularly, in-home networks have also increased in heterogeneity by using various devices such as laptops, cell phone and PDAs. In our study, a set-top box(STB) is the access pointer between the internet and a home network. We design an *ActiveSTB* which has a capability of buffering and quality adaptation based on the estimation for the available bandwidth in the wireless LAN.

To My Family

ACKNOWLEDGMENTS

First of all, I offer my sincerest gratitude to my supervisor, Eun Jun Kim. She encouraged me in challenging new problems and finding the solutions. I also appreciate my committee members, Dr. Riccardo Bettati, Dr. Rabi N. Mahapatra, Dr. Deepa Kundur and Dr. Ki Hwan Yum, for supporting me as I completed my research.

I also appreciate my friends, colleagues, and department faculty and staff for my experience at Texas A&M University. To design efficient simulation for web service, I was supported by Gopinath Vageesan who is very kind and smart. I also appreciate Baik Song An who had a discussion on streaming service with me. In addition, I got help from Dr. Pyoung Soo Mah and Kyoung Ill Kim from Electronics and Telecommunications Research Institute. I have been blessed to have worked with friendly members.

I appreciate my parents for supporting me throughout all my studies at Texas A&M University. Finally, I give my credit my wife, So Jong Kang, and my daughter, Hae Jee Lee. Without their support, I could not have completed my work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Cluster systems have been increasingly applied to web servers, file distribution and database transactions for their load sharing and high-performance capabilities. Commercial servers like Google configure thousands of PCs (about 15,000 nodes) to handle the high volume of traffic. The design of distributed web servers has been a major research thrust to improve response times. Web clients still experience long and unpredictable delays when retrieving web pages from the cluster systems. It has been observed that web servers contribute to approximately 40 percent of the overall delay [1], and this delay is likely to grow with the increasing use of dynamic contents. Web cluster systems incur an additional delay in analyzing the incoming request and forwarding the request to one of the backend servers. Therefore, the delay at the web server is a critical component which has to be reduced to achieve performance enhancement of web cluster systems.

For reducing the delay, previous studies can be classified into two groups: system management [2, 3, 4, 5, 6, 7] and service design [8, 9, 10, 11]. In the system management, it is critical to decide which node processes incoming requests. Each backend server has a different status on memory usage and CPU workload based on previous requests. Therefore, the design of distribution determines the overall delay in web cluster systems. We propose an efficient distribution scheme through grouping web requests. Other approaches in the system management reduce the delay through preprocessing incoming requests. Preparing incoming requests achieve the performance enhancement without new equipment. In this study, a proactive

The journal model is *IEEE Transactions on Automatic Control.*

web processing scheme is suggested using the relation between web requests. Service design manages the heavy-loaded services such as web streaming. Heavy loaded services consume a lot of memory and available bandwidth in the network. The inefficient web service wastes the system resources and increases the delay. We propose an estimation scheme for available resource and a management scheme for removal of unnecessary system efforts.

In this study, we explore web cluster designs in three domains and investigate the scheme for performance enhancement with each domain. One domain is the design of request distribution. Among the different architectures in the cluster-based servers, the distributor-based systems have been widely deployed as shown in Fig. 1. The distributors determine the overall performance in web cluster systems. Another domain is the prefetch scheme in backend servers. In Fig. 1, backend servers are located behind distributor and process the incoming requests. The last domain is efficient web streaming. Recent web sites provide streaming service to web users. It is critical to know how much streaming data can be transmitted. This dissertation includes the design studies for web cluster in the following directions.

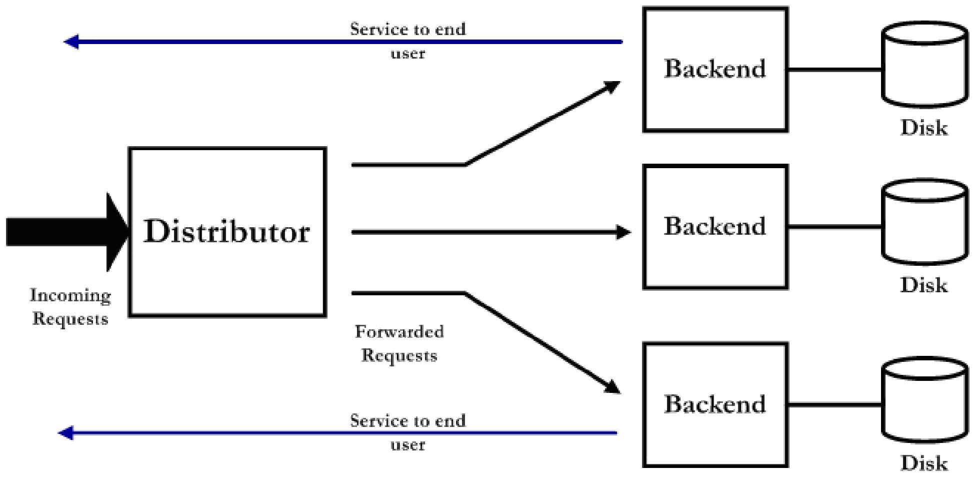

Fig. 1. Distributor-based Web Cluster System

A. Web Request Distribution

In locality-based request distribution schemes, the distributor contacts the dispatcher to obtain the locality information. If the page is located in the same backend server, the request is serviced directly. Otherwise, the distributor forwards the request to the backend server that has a better locality for the file. The role of the dispatcher is to specify the locality of the requested files to the distributor. The forwarding of the requests from the distributor to the backend servers is carried out in complete transparency to the users. A handoff protocol and TCP splicing are employed in most cases to make the transition smooth and transparent [3, 12, 13]. The requests are forwarded to the set of backend servers based on a certain policy. LARD (Locality Aware Request Distribution) [3], PARD (Power Aware Request Distribution) [14] and WRR (Weighted Round Robin) are a few of the most prolifically adopted policies. The policies focus on improving efficiency, power conservation and load balancing, respectively.

In this study, we present a new proactive request distribution scheme (*ProRD*), which reduces the delay at the web server by bundling requests. We improve the distribution policy by dispatching the requests based on the analysis of the log files of web servers. The general user navigation pattern, user behavior and general website organization are some of the critical information that is extracted from the web server log files. This information is made available for the distributor to discern and classify the incoming requests and perform the dispatch to the appropriate backend server. Also, our *ProRD* provides memory aware request distribution which distributes incoming requests to the prefetch-enabled backend servers efficiently. Locality-based distribution policy is commonly accepted to improve the performance using the locality of incoming requests. However, it may choose which non-uniform distribution can use up the memory at the particular backend server. Our *ProRD* reduces the performance drop with the efficient memory management at cluster environments.

B. Web Prefetch Scheme

To solve the delay problem of the web cluster system, there have been many studies [15, 16, 17, 18] on the main memory management such as web cache schemes and web prefetch schemes. Although web cache schemes reduce the network and I/O bandwidth consumption, they still suffer from a low hit rate, stale data and inefficient resource management. [19] shows that an inefficient web cache management, also called *the Slashdot effect* caused the crash of a major news web site. In addition, the frequent update of web objects reduces the hit rate and disturbs the efficient web cache management. Although web cache space is unlimited, it cannot update the large volume of web objects properly.

Web prefetch schemes overcome the limitation of web cache mechanisms through pre-processing contents before a user request comes. Web prefetch schemes predict future requests through web log file analysis and prepare the expected requests before receiving web objects. By relocating the expected web data into the memory, it can achieve efficient resource management and increase the hit rate on memory. Compared with web cache schemes, web prefetch schemes focus on the spatial locality of objects when current requests are related with previous requests. Web prefetch schemes increase the bandwidth utilization and reduce or hide the latency due to bottleneck at the web server. However, despite these benefits, three difficulties prevent prefetch schemes from being exploited in web cluster systems. First, it is difficult to find which objects are related with the incoming requests. At the server side, web access patterns are difficult to predict because of the web cache mechanism. Second, it is difficult to find an optimal prefetch rate. Too aggressive prefetch schemes may hurt overall performance due to the shortage of memory. Furthermore, in modern web frameworks, there is a trade-off between consuming available memory space and increasing the performance of web systems. Persistent HTTP requires the dedicated memory for web server processes even in idle time. Also, allowing multiple connections per client may lead to the shortage of memory. Finally, a prefetch scheme in a web cluster system should be considered along with an efficient resource management. Inappropriate resource management drains the resource of one backend server, while other backend servers have the available resource.

To overcome these difficulties, we design adaptive web prefetch schemes for web cluster environments. Our prefetch schemes decide which web objects are to be prefetched by considering memory status of the web cluster system. When processing the incoming request, backend servers prefetch related web objects depending on the local memory situation and report the list of the prefetched web objects to the web distributor. Our adaptive scheme consists of two components: Double Prediction-by-Partial-Match Scheme (*DPS*) and Adaptive Rate Controller (*ARC*). First, we design a dynamic web prediction scheme called Double Prediction-by-Partial-Match Scheme (*DPS*). Web access patterns in web systems are dynamic depending on the location of a client. When web objects are stored in an intermediate node, requests to those cached objects do not reach the web server. Also, persistent HTTP and multiple connections from a client eliminate time intervals between consecutive requests which give the opportunity to prefetch web objects. In addition, the performance of the web prefetch scheme is dependent on the workload [20]. The *DPS* scheme solves the problem by providing the adaptiveness that handles the client's random access pattern.

Second, we suggest Adaptive Rate Controller (*ARC*) that provides an adaptive prefetch rate at run time. In multiprocessing environments, web processes allocate memory by their needs. In modern web frameworks, persistent HTTP improves the performance through avoiding the frequent network connection establishment and web server process creation. However, it requires the dedicated memory for web server processes even in idle time. Also, a modern web client uses multiple connections to improve the web throughput. However, it consumes a lot of memory of web servers because several web processes are dedicated at one client. We cannot provide the system with unlimited memory, therefore aggressive prefetch schemes can interfere with demand requests from the same client or other clients. For improving the performance of prefetch schemes, the *ARC* scheme prefetches web objects depending on the memory status.

Recent works [21, 22, 23, 18] show that the unorganized and simply aggressive web prefetch schemes cannot improve performance and even increase the processing time of user request. Aggressive prefetch schemes also consume the memory for prefetching useless objects at the selected server. It can cause the delay in the overall web cluster system. Our schemes avoid the skewed distribution of requests in the web cluster system.

C. Web Streaming Service

This research provides a solution to networking issues related to web streaming service in the wireless home environment. Competitive pricing of home-based network electronic devices has caused the home network to rapidly increase in complexity. Home networks consist of various network devices from multiple vendors and different hardware generations that are added to the home over time. Also internet access in home environments has significantly increased and is deeply heterogeneous.

The rapid and widespread usage of the internet has given rise to an increase in demand for web streaming service. There has been significant increase in user requests for streaming audio/visual information over the internet in order to be accepted as an alternative by the mass television audiences. In addition, Wireless Local Area Networks (WLANs) make multimedia streams commonplace, and terminals are diversifying into hand-held devices such as personal digital assistants (PDAs), laptops and audio/video players. These heterogeneous devices have different access patterns and mobility [24].

Heterogenous multimedia streaming service has led to many technical challenges that must be addressed in the two areas of video coding and networking. One such method that addresses the challenge associated with multimedia streaming and networking is scalability. Scalability plays a crucial role in delivering the best possible video quality over various environments [25, 26]. Even though multimedia scalable streaming service provides various rate multimedia contents, there are still problems in estimating the point in time to change the bit rate of the transmitted bit stream.

Estimating the available network bandwidth in a WLAN is very challenging and crucial for multimedia streaming services. Although there can be various wireless environments where multimedia streaming services are provided, we mainly focus on the LAN/WAN shown in Fig. 2. In this figure, an Internet-based Set Top Box (STB) is the interface between a wired network and a wireless network. STBs receive the television signal, run the interactive applications and transfer the digital TV signal to the TV. STBs are becoming key devices in home entertainment networks, not only to receive digital television, but also as a residential gateway to deliver multiple services as well. STBs in home networks have gained in flexibility and modularity, therefore, the functionality of the STB may be distributed between a main device and several peripherals, all interconnected by an Ethernet or wireless network.

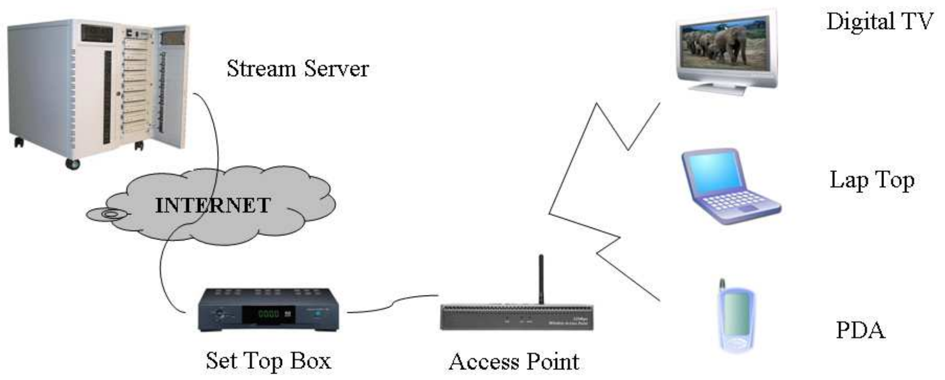

Fig. 2. Streaming Service in Wireless LAN

In scalable multimedia streaming services, it is very critical for the STB to know the available wireless network bandwidth. The STB serves as a bottleneck for the server and heterogeneous client devices, and the bottleneck determines the quality of a delivered multimedia to each end-user. In a wireless network, the IEEE 802.11 protocol in Distributed Co-ordination Function (DCF) mode, based on CSMA/CA algorithm, is becoming very popular. Previous works [11, 27, 28] based on the bandwidth estimation of wired environments are not applicable to wireless networks that use the DCF protocol. Multimedia streaming is a soft real-time service where each frame is delay-sensitive. Swiftness and availability is critical for real time systems. During bandwidth deviations, the rate of the transmitted multimedia streams should be changed expeditiously. The accuracy of previous works, Spruce [11] and ProbeGap [28], is dependent on probing time and the volume of the packets for probing. ProbeGap produces good estimates at low cross traffic rates (2 Mbps cross traffic regardless of the cross traffic packet size); however, it significantly overestimates available bandwidth when the cross traffic is high (4 Mbps cross traffic generated with 300-byte packets) [28]. Influence by cross traffic on probe packet sequences causes probe packets in sequences to be split up or even lost. Our contribution in this study is two-fold. First, we suggest *IdleGap*, which is a bandwidth estimation tool for a real-time system in a wireless network and is independent of cross traffic. Second, our STB, *ActiveSTB*, will be to forward a partially buffered stream to a client to allow a user to view quality playback of video, and additionally simultaneously perform quality-adaptation to change in network bandwidth.

This dissertation is organized as follows. Chapter II explains the *ProRD* scheme for enhancing the distribution policy at distributor. In Chapter III, we present the prefetch designs in web cluster environments based on web access patterns. Chapter IV presents some issues in web streaming scheme through wireless networks, followed by the conclusion in Chapter V.

CHAPTER II

REQUEST DISTRIBUTION

Distributor decides the overall performance of web cluster systems. Many studies [3, 29, 14] provide the solution on performance enhancement. However, they did not consider about the structure of web documents and the relation between requests.

A. Background

Of the various policies that are employed at the distributor, the policies that provide better load balancing among the backend servers, better efficiency and considerable power conservation are the most preferred. In this chapter, we would review the existing distribution schemes including weighted round robin, locality aware request distribution and power aware request distribution.

1. Weighted Round Robin

The choice of the policy is very critical for the efficient operation of the system. The weighted round robin policy is applied based on the current load at the backend servers. The policy is applied at the distributor, where the requests are forwarded to the backend servers. The distributor maintains the record of the current load at the backend servers and it forwards the request from the client, based on this information. The request is forwarded to the least loaded backend server among the bunch of servers. The request forwarding is thus weighted based on the current load on the servers. The server that is most loaded is relieved off the load by forwarding the requests to the least loaded server. So, at any given point of time, the load is evenly balanced among all the available servers and thus providing very good load balancing.

The main drawback of the system is that it does not concern about the locality of the requests As the system does not consider the locality of the data among the backend servers, the even requests to same objects land up in different servers and incur large disk latencies. This increases the response time of the servers and hence the throughput. Considering this, power and locality based request distribution policies have more significance.

2. Locality Aware Request Distribution

The major drawback of the weighted round robin is that it incurs a large amount of disk latency by not considering the locality of the data in the backend server memory. Our simulation of the weighted round robin shows that during worst case, this could generate an unacceptable amount of disk latency in the backend server and lead to increased response time of the server and reduced throughput. This causes large delays to be experienced by the clients and brings down the performance of the whole system. To overcome this, locality aware request distribution [3, 29] employs locality based distribution policy at the distributor and strives to increase the memory hits at the backend server rather than the disk latency.

The distributor maintains the table of previous request distribution. When a new request arrives at the distributor, requested object is looked up in the distributor table and the corresponding server is identified. The request is forwarded always to that server for that particular objects. By this assignment, the request will incur disk latency only during the first initial assignment to that backend server. The consecutive requests of the same object end up as server memory hits, since it has already been fetched from the disk and is now in the memory. Once the requests starts overflowing at one of the servers, one of the least loaded servers is added to serve the requested object and the server set for the requested object starts growing.

Similarly, when a server becomes underutilized, a server is removed from the server set.

But, the major drawback of the locality aware request distribution system is that it too does not take into consideration about the multiple connections from a client and persistent HTTP. This incurs the memory shortage of the some backend servers, even though other backend servers have enough available memory.

3. Power Aware Request Distribution

Power aware request distribution takes a great consideration to reduce power consumption of the cluster system, and hence, power aware request distribution [14] is the most efficient policy in terms of power saving among the three policies. Power aware request distribution employs the ON-OFF Model; any backend server, which is idle, is turned off, and backend servers are turned on whenever they are required to serve requests. In [14], they assume that power is equal to its maximum power when it is ON and simply zero when it is OFF. Turning off unused server is considered to be the best way to save power. However, this policy brings more number of disk latency than locality aware request distribution scheme. Whenever backend servers are turned ON, the data stored in the backend server memory is completely deleted. This causes a lot of disk latency as distributor forwards requests to the backend servers. Furthermore, there are startup delay and shutdown delay when backend servers are turned ON and OFF, respectively. While the startup delay results during the booting up of the operating System, the shutdown delay is a period between pruning the idle backend server from service mode and shutting it down. E. V. Carrera et al., [2] have proposed a technique to conserve energy in network servers using a multi-speed disk technology. The idea is to use two disks with different speed to emulate a multi-speed disk. Thus multi-speed disk concepts need to be applied to achieve this kind of optimization. When the load is hits higher than a pre-defined threshold, the disk with fast response time serves it, and vice versa. E. Pinheiro et al., [30] have proposed a dynamic cluster reconfiguration technique., to bring down the energy consumption in servers. In this technique, a cluster node is dynamically added or removed to the cluster system based on the their purpose.

B. Motivation

1. Distributor Based Web Cluster System

Among different architectures in the web cluster system, the distributor-based systems have been widely adopted as shown in Fig. 1. These systems have a web distributor that forwards the requests to any of the backend servers. The web distributor selects the backend server based on its own policy. Forwarding of the requests from the distributor to the backend servers is carried out in complete transparency with the web users using Handoff protocol and TCP splicing scheme. Locality Aware Request Distribution (LARD) [3, 29], Power Aware Request Distribution (PARD) [14] and Weighted Round Robin (WRR) are a few of the most prolifically adopted policies, which focus on improving efficiency, power conservation and load balance respectively.

Locality-based request distribution forms the heart of the system. The policy that we use mimics the LARD [3, 29] system. The distributor reads the header of the incoming requests and looks up the distributor table to determine the locality of the file among the backend servers. Once a backend server is chosen, the distributor forwards the request to that backend server and hands-off the connection. The backend server then replies back to the requesting client with the requested file. The choice of the backend server depends on the lookup at the distributor table and the current load conditions at the server. The algorithm uses the load thresholds for determining the load conditions at the backend server. $Load_{low}$ stands for the lower load threshold, below which the server is considered to be underutilized. Similarly, $Load_{high}$ stands for the higher load threshold, above which the server is considered to be heavily loaded. It increases the locality on each backend server.

On power conservation standpoint, many researchers have previously proposed schemes to tackle power consumption. E. Pinheiro et al., [30] proposed a dynamic cluster reconfiguration technique to bring down the power consumption in servers. In Power-Aware Request Distribution (PARD) [14], a simple power scheme was adopted to conserve power. It used a simple ON-OFF Model, where any backend server, which was idle, was turned OFF. And backend servers were turned ON whenever the demand for service increased. However, this policy suffered from a performance viewpoint. PARD does not have performance enhancement measures and also reduced the locality of the files by wiping off the contents of the memory which is being turned OFF. Startup delay and cold miss of a server to turn ON from OFF state could incur the degradation of the performance at web cluster system.

2. HTTP 1.0 vs HTTP 1.1

When web client requests a HTML document, the requests to main object and its embedded objects are sending to web server sequentially. For releasing the overhead on web transactions, HTTP 1.1 provides two schemes including persistent-HTTP and web pipeline as default. In HTTP 1.0, each request requires the separate TCP connections, while multiple web requests over HTTP 1.1 can be processed at a time. In addition, HTTP 1.1 save the workload to manage the TCP connections in web hosts such as clients, proxies and servers. However, HTTP 1.1 makes it difficult for web distributor to read each requested file. Web distributor can read only the first request and distribute it with locality. Other following requests should be distributed into the backend server to take care of the first request. These following requests over HTTP 1.1 drop the locality in the web cluster system which increased the response time for the requests. The enhancement of the locality of the forwarded requests save the response time in web cluster systems. Also, prefetching and data placement at the backend server enhance the performance of web cluster systems. In this study, we propose a new proactive request distribution (*ProRD*) scheme, which reduces the delay at cluster-based web server by bundling requests and prefetching them.

3. Layer-4 Switches and Layer-7 Switches

In the web cluster system, each node has their own address and process the forwarded requests and web switch can access the each node using their address. Web switch decides the overall performance in web cluster system. Therefore, various scheme are suggested to improve the performance in web cluster system. We can classify the switch techniques into two groups based on protocol stacks where web switch decides which backend nodes processes the incoming requests. First group is that web switch distributed the incoming requests in the layer 4, while other groups decides the backend node to process the incoming requests in the layer 7. Two schemes provide the different switching schemes based on the different policy.

In the Layer-4 scheme, web switch preform web switch with the ignorance about the requested web objects. When clients send requests into the web server, they should establish the TCP connection. When web switch receives the TCP SYN packet from clients, web switch decides which backend server processes the incoming requests. It can save the distribution time in web switch, because web switch do not need to access the contests. However, they cannot provide the intelligent distribution depending on the requested objects.

After extracting the name of requested objects, Layer-7 web switch decides which backend nodes process the incoming requests. Therefore, web switch should process the TCP connection from clients, and establish the connection to the backend servers. Web switch cannot provide the fast distribution such as Layer-4 web switch. It takes time to process TCP connection and decode the HTTP packets from clients. However, Layer-7 switch can provide the intelligent distribution based on the requested objects. They can increase the locality of requested file.

The main difference of two schemes are the connection management. In the layer-4 scheme, web switch just forwards the packet through one TCP connection such as router. Therefore, packets from clients can send to the backend server directly, and also backend server can reply to the client without the assistance. However, Layer-7 switch provides the separate TCP connections between client and server. The packets from clients are decoded by Layer-7 web switch and web switch wraps it following by the HTTP protocol stack.

4. Web Server on Kernel Mode

For minimizing the processed event, web server was working in the kernel mode. In the general web system, kernel detected the receiving event from network device. After completion of three-way TCP handshake, kernel received the data with user request then copy received data to user area. Besides, process and response of end-user's request lead the transition between kernel mode and user mode. For saving the switch time between kernel mode and user mode, web server cached web contents on the kernel mode [31, 32]. Scalable Web Cache (SWC) [31] working on windows 2000 TCP/IP, while kHTTPd [32] and Tux [1] handled socket interface in Linux. kHTTPd could not handle dynamic contents, so it needed the back up server to handle dynamic contents. TUX was the integrated web server in operating system and could handle even dynamic contents. In [33], even though the rate of dynamic contents and static contents in the workload determined the speed of request, TUX outperformed other web servers in static contents.

C. Proactive Request Distribution (*ProRD*)

Layer-7 web switch scheme provide the intelligent distribution scheme. But, it suffers from the distribution delay. Our *ProRD* scheme save the distribution time while it provides the efficient distribution using web structures.

1. Distribution Policy

Our *ProRD* scheme provides two policy to improve the performance of the web cluster systems. First of all, a *ProRD* classifies the incoming requests into groups including main request and sub requests. And it provides the selective distribution. Secondly, they provides the memory aware request distribution to avoid the creation of new web processes.

a. Group Request Distribution

From our observation, web clients send requests for embedded objects and expected web pages immediately after the requests to main object are replied from the backend server. To release the overhead at the web distributor, *ProRD* distributes the bundle of requests into the same backend server. In general, the web log files contain the history of user requests that let us know the relations between the main page and the embedded object, the user navigation patterns and the classification of web pages according to user requests. Based on the extracted information, the web distributor in a *ProRD* scheme can distribute a bundle of requests, rather than only one request.

Like the locality distribution scheme [3, 29], our *ProRD* scheme stores the distribution information in their own memory. Also, our *ProRD* scheme distributes the requests into the backend server that processed the previous requests to the same objects. This scheme can increases the locality using the distribution information. When requests are distributed, web switch extracts the requested object and search the requested object in the the table for distribution information. It takes time to search the previous distribution information and re-establish TCP connection to backend server in the cluster. *ProRD* avoids the full distribution to the whole requests from client. The embedded objects are followed by the main object, therefore *ProRD* distributes the only main objects and just forwards the embedded objects to the backend server that process the main object.

When backend server provide the prefetch scheme to decrease the response time, *ProRD* scheme receives the prefetched objects from backend server and stores the information. It forwards the requests to the backend server that prefetched the requested objects based on the stored information. If distributor does not provide the prefetched information. Prefetch scheme in backend server leads the miss to increasing response time and wasting the bandwidth between memory and disk, the *ProRD* scheme can save response time to web users in prefetch hit.

In Fig. 3, when HTML document A is requested on the established persistent HTTP, the embedded objects in HTML document A are prefetched to the memory on the backend server. It saves the response time the following requests to embedded objects. When requests to embedded objects are incoming, the backend server then replies to web user based on prefetched embedded objects. As web objects are prefetched on the prefetch memory of the backend servers, the list of the prefetched files are updated at the distributor. When the load of the server exceeds threshold, a *ProRD* selects a new backend server for the migration of the prefetched web objects from the heavy-loaded backend.

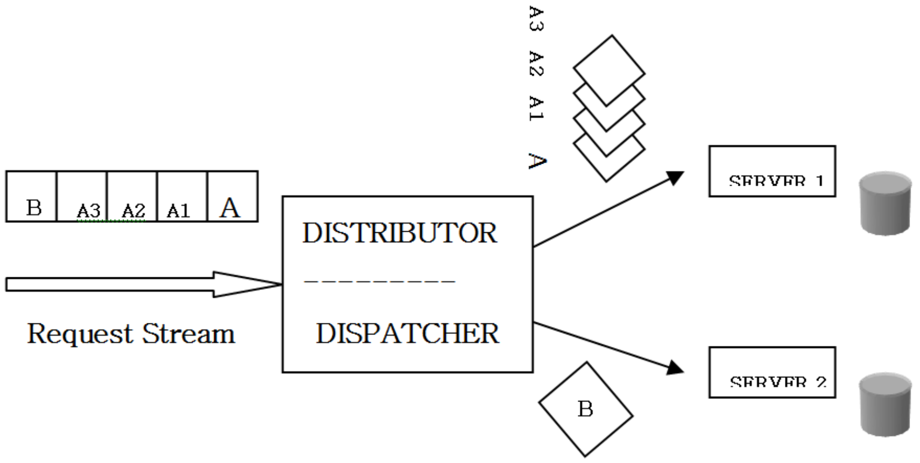

Fig. 3. Distribution in *ProRD*

b. Memory Aware Request Distribution (*MARD*)

As in Fig. 4, there are two groups of nodes in a web cluster system; web distributor and backend nodes. A web distributor forwards incoming requests to backend servers based on its policy. In our simulation, a distributor uses a locality-based distribution and TCP splicing scheme.

In process-based web servers such as Apache or ISS, there are several idle processes waiting for new clients. When there are inefficient idle processes, the web server creates a new web process and consumes the memory space which could be used to cache file otherwise. Therefore, unnecessary web processes decrease the benefit of the buffer cache in the web server. To avoid the skewed distribution, a *ProRD* checks the number of idle web processes of backend servers. Our *ProRD* does not forward a new request to those with inefficient idle process. A *ProRD* prevents the memory shortage due to too many web processes that decrease the cached or prefetched web objects.

Web distributor updates the distribution table after distribution of the incoming request and the acquisition of prefetch information. When distributor has the history information or prefetch information, the requests are forwarded into the connected backend server that satisfy the load and the number of connections. If not, distributor selects the optimal backend server for incoming request and store distribution history into distribution table for next decision.

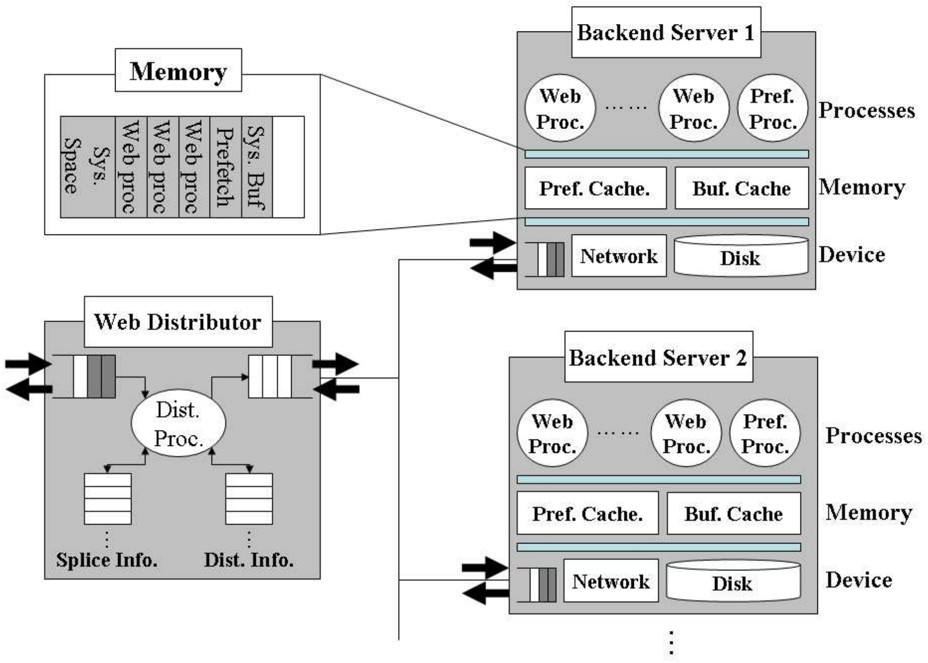

Fig. 4. Web Prefetch Cluster System

2. Diagram of Distribution

Because the distributor of a web cluster system is the bottleneck of the system, an efficient distribution policy is critical to improve the performance of the system. In a *ProRD*, the distribution policy is composed of several steps. In Fig. 5, the white boxes show the procedure for previous LARD schemes. The light grey boxes and dark grey box are for *ProRD without MARD* and *ProRD with MARD*, respectively.

First, the distributor reads the incoming request. Fig. 5 illustrates the steps involved in distributing the incoming requests. The first step is shown as 'read the incoming request' process in Fig. 5. At the second step, *ProRD* classify the requests into two groups including main objects and embedded objects. When the incoming requests are for the embedded objects and already establish TCP connection with one backend server, the requests are just forwarding to the already selected backend server. The distributor selects a backend server based on a record of prefetched objects and previous distributions at the third step. Before forwarding the incoming requests, they checks the selected backend servers. When the number of connections exceeds the threshold or number of the idle web processes are not enough, *ProRD* finds the other backend servers. Finally, a *ProRD* does not find the backend server to process the incoming requests in the previous steps, it checks whether the request is the first request or not after TCP establishment. When the request is not first, the incoming request is just forwarded into the already selected backend server. If not, a *ProRD* distributes the incoming request into the least loaded backend server. When the content is not available on the memory of the backend server, the requests incur misses.

Our *ProRD* avoids the shortage of the memory, then increases the hit rate on the memory and decreases the number of dispatch hits dramatically and improves the performance of the cluster system.

D. Experimental Results

The simulation model consists of a distributor and backend servers. Our model is scalable to any number of backend servers and we show that results are consistent with 4 to 12 backend servers. The model emulates a real-time cluster system with

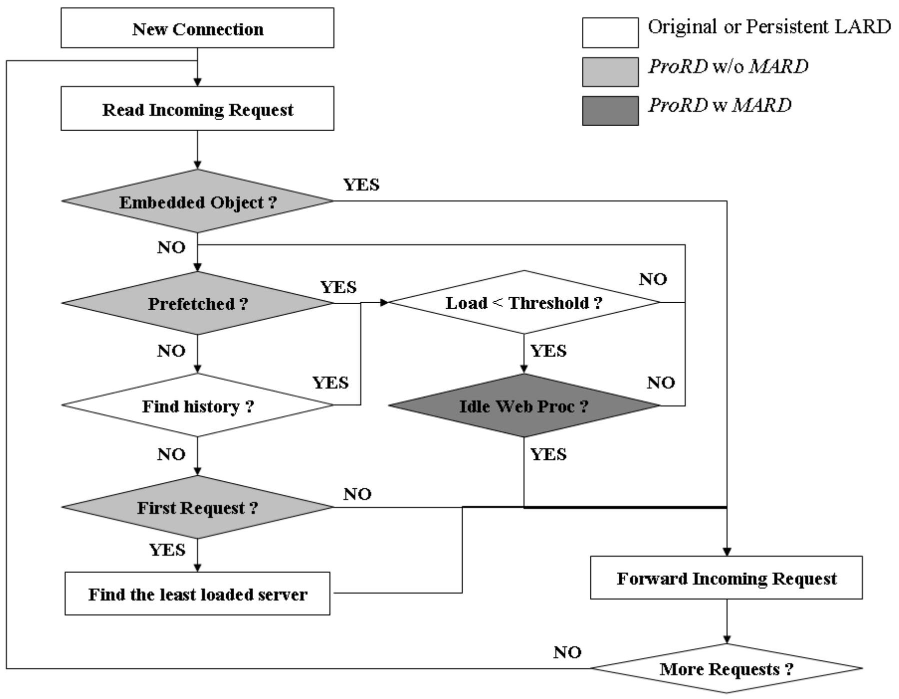

Fig. 5. Flow Chart for Distribution in *ProRD*

request queues at the distributor and the backend servers.

1. Switch Time Measurement

For estimation of the elapsed time for web switching, we modify the real web switch program, 'Layer-7 Web Switch'. When the incoming requests are coming into the web cluster, web switch process it following the one of four cases including 'switch hit', 'switch miss but reuse', 'switch miss and reconnect' and 'switch forward'. When the web switch find the previous switch information then just forwards the incoming requests to the backend server followed by previous information. In 'switch miss but reuse' and 'switch miss and reconnect', web switch should re-distribute the incoming requests into other backend server at even the 'switch hit'. In the 'switch miss but reuse', web switch selects the previous backend server, so they just reuse the TCP connections with backend server. However, web switch selects the other backend server at the 'switch miss and reconnect' and requires the new TCP connection with the selected backend server. The 'switch forward' is for the 'Layer-4 web switch'.

Table I. Switch Elapsed Time

Switch Hit	Switch Miss But Reuse	Switch Miss And Reconnect	Switch Forward
90.444 MS	92.088 MS	2362.927 MS	40 MS

Table I shows the elapsed time for web switch. We install the simple client and modified web switch at the machine and forward the incoming requests into other computer. Computer has Intel Core 2 duo processors and two GB memory. The whole computers are connected by the 100 Mbps Ethernet.

A simple web client sends the requests to the web switch, then web switch forwards the incoming requests into backend server. After receiving it, the backend server replies the dummy packet into the web switch. Finally, web switch forwards the packets from web server into simple web clients. We test 1000 requests using the implemented machine at each case. The 'Switch Forward' provides the web switch without analysing web contents. And, the 'Switch Forward' can save the contents delivery between kernel and user. The elapsed time of 'Switch Miss But Reuse' added the time for the distribution decision and the elapsed time of 'Switch Hit'. But, the decision time is not distinguishable. However, it takes time to re-establish TCP connections with other backend server.

2. Simulation

Simulations have been carried out by implementing the proposed algorithms using the CSim [36] and DiskSim [34]. The program is a scalable, user configurable cluster with realistic system and disk queues. Additionally, we have implemented the round robin

Table II. Simulation Parameters (Request Distribution Simulation)

Backend Server	
Parameter	Value
Physical Memory	512 MB
Kernel Space	100 MB
Web Process Size	20 MB
Disk Latency	Provided by DiskSim [34]
Memory Latency (Hit)	500 Mbps
Network Latency	100 Mbps
Web Distributor	
Switch Hit/miss/forward	in Table I
Session Time	15 Seconds [35]
Web Client	
Concurrent Connection	Up to 4

scheme, the original LARD scheme, the persistent LARD, the *ProRD without MARD* scheme and a *ProRD with MARD* scheme. The simulation code emulates a cluster system, which takes any log file in common log format as the input. The log files used for the simulations are the request logs to the Texas A&M University CS department website, NASA and ClarkNet log files. The following metrics are closely monitored for evaluating the performance of the system: Average Response Time, Hit Rate and Access Time. We compare our policy, *ProRD with MARD* and *ProRD without MARD*, against the existing distribution scheme including round robin, the original LARD and the persistent LARD. The original LARD and the persistent LARD are suggested for increasing the locality, while we design *ProRD without MARD* that avoids the web process creation.

Table II shows the configuration in the simulation for evaluating the *ProRD* system. The physical memory in the system is 512 MB. Kernel consumes the 100 MB in the memory. We assume that web process is created at each incoming request and requires 20 MB. We embedded the 'DiskSim' simulator in our simulator to configure the disk access time. It takes 500 Mbps for web process to access a page in the memory, while transmission speed of the LAN is 100 Mbps. The connection between web client and server has sustained 15 seconds without web transaction. Also, web client can establish multiple connections with web server. We assume that web client establish four connections with web server.

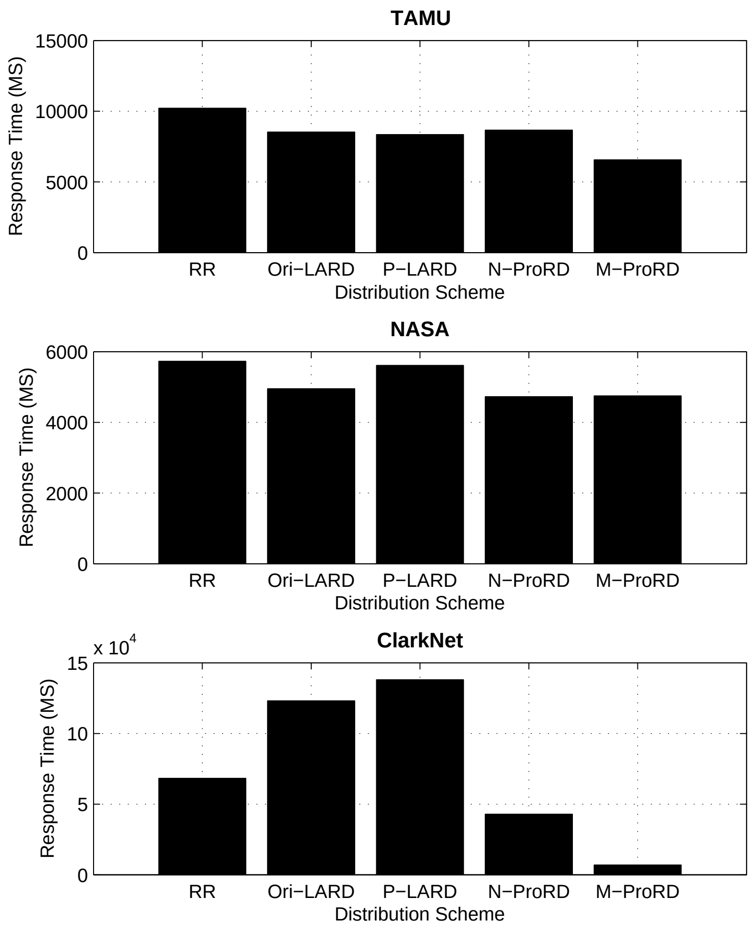

Fig. 6. Response Time under Various Distribution Schemes

Fig. 6 shows the response time under distribution scheme. X axes show the distribution scheme and Y axes stand for the response time. In the X axes, the 'RR', the 'Ori-LARD' and the 'P-LARD' are the existing distribution schemes and show round robin, original LARD and persistent LARD, respectively. '*N-ProRD*' and '*M-ProRD*' stands for *ProRD without MARD* and *ProRD with MARD*. Even though the round robin scheme provides the fast distribution to the incoming requests, the overall response time is increasing. It does not care the requested contents and lose the locality of cluster system. The original LARD and the persistent LARD distribution scheme provide the short response time, because they provide the high locality for requests distribution. However, it can cause the unequal load balance and the shortage of the memory. Finally, it loses the data from the memory. Even though *ProRD without MARD* increases the performance for avoiding the unnecessary web process creation. Our *ProRD with MARD* provides the fast distribution and high locality.

In Fig. 7, *ProRD* and extended LARD schemes show the high hit on the memory. Both of them can provides the high locality that increase the number of hit. But, the extended LARD schemes checks the every requests from client. It can make the overhead on the distributor. The persistent-HTTP reuse the TCP connections, therefore client can send several requests into web cluster. The extended LARD should distribute all of them one by one. The round robin scheme loses the chance to improve the performance because they do not care about the requested contents. The original LARD and persistent LARD scheme increases the unequal distribution of the incoming requests and cause the shortage of the memory.

Fig. 8 shows standard deviation of the number of the requests that are processed at backend server. To process the incoming request, the round robin scheme just selects the node one by one. But, the standard deviate of the round-robin scheme

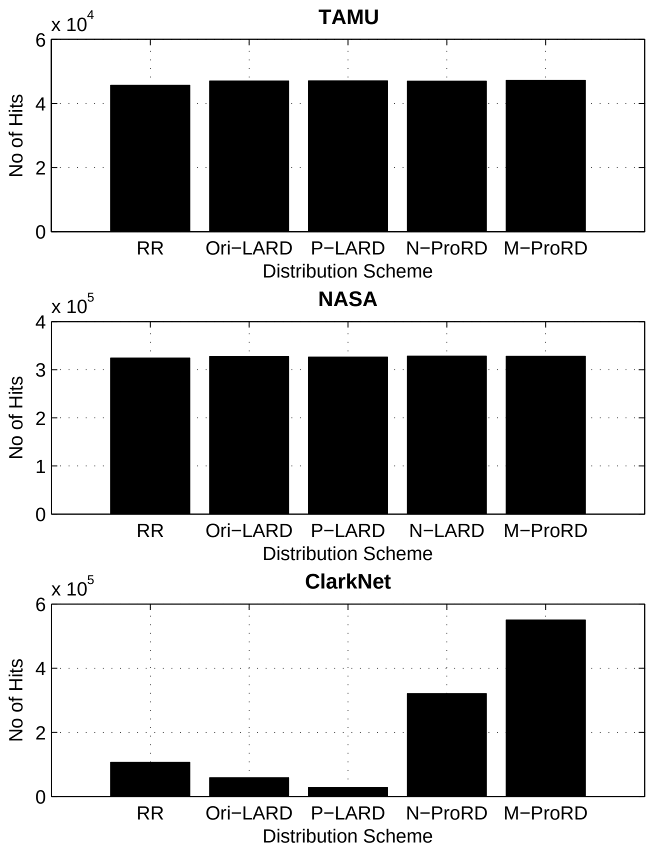

Fig. 7. Number of Hits

is not zero. The reason is that the client using the persistent-HTTP clients can send the different number of requests in one TCP connection. The original LARD and the persistent LARD increase the unequal distribution. The persistent LARD re-distribute the every requests, so it increase the unequal distribution. Our *ProRD* shows the low standard deviation that shows the good load-balance in the web cluster system.

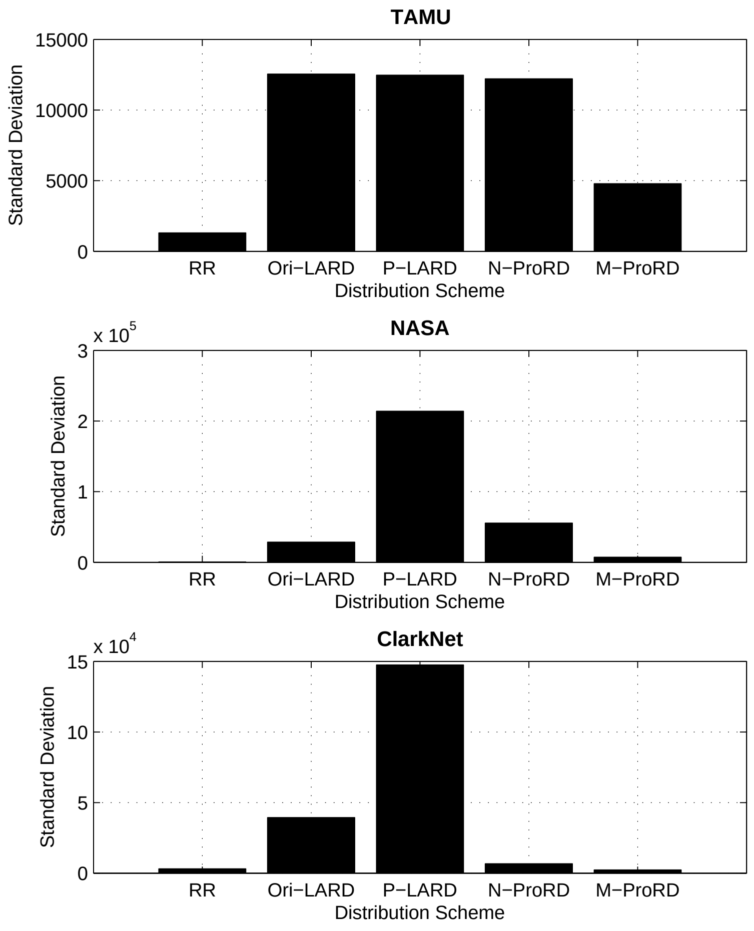

Fig. 8. Load Balance of the Incoming Requests

CHAPTER III

WEB PREFETCH SCHEME

A. Background

Prefetch scheme is used in various area for performance enhancement of the web cluster system. Especially, the performance of web systems are improved through web prefetch scheme. Prefetch schemes are composed of two sub schemes including prediction schemes and management schemes.

1. Web Prefetch Schemes

Web prefetch schemes are roughly classified into two groups; short-term prefetch schemes and long-term prefetch schemes. Short-term prefetch schemes predict future requests based on the recent historical information. [37] proposes the prefetch algorithm for a general file system. [38, 39] predict the next incoming requests using the N-th order Markov models. [39] suggests heuristic schemes to reduce high complexity at a multi-level Markov models.

The Prediction-by-Partial-Match (P.P.M.) models complement N-th Markov model. The order of the Markov model increases not only the accuracy but also the complexity at the same time. P.P.M. schemes predict the future incoming requests using the top-n related objects [4, 40, 6] or objects with confident threshold [41, 42, 43]. In [40], they search the long sequences for frequently accessed patterns. [44] suggests dynamic P.P.M. models. The recent research evaluates the prediction schemes [7, 20].

The long-term prefetch scheme defines clusters of web objects using access patterns, then prefetches web objects in the unit of a cluster. [45, 46] provide replacement policies for Content Distribution Network (CDN) platform, while [47, 48] suggest the replacement algorithm for mobile environments. [49] provides a hierarchical clustering system that groups search results into several folders. In [50], a divide-and-merge scheme creates the cluster of web objects by combinational approaches between top-down and bottom-up schemes. [51] suggests the modified proxy model to prefetch the embedded objects from the web server. [52] provides a web cluster scheme using a vector model and semantic power.

Hybrid prefetch schemes integrate the short-term prefetch based on the Markov model and the long-term prefetch using cluster scheme. [53] suggests the combinational prediction scheme of existing models including Markov models, sequential association rules, association rules and cluster schemes. [54] generates *Significant Usage Patterns* based on abstraction techniques and also provides the path between *Significant Usage Patterns* using the Markov model. [55, 56, 57] create the cluster based on their policy including *Expectation-Maximization* [56], *CitationCluster* [55] or K-means cluster scheme [57]. After generating the cluster, they use the Markov model to find the relationship between clusters of web objects. Although the hybrid scheme provides the benefit of both short-term and long-term schemes, it does not support the memory-consuming modern web frameworks.

2. System Prefetch Schemes

System prefetching techniques are widely studied for improving the system performance including system design approaches, compiler-aided approaches and dynamic approaches.

System design approaches improve the performance of memory management through prefetching scheme. [58] analyzes the workload on the virtual channels and suggests the management scheme for virtual channel buffers, while [15] provides the management framework for prefetch that divides the main system memory. [59] provides the two separate queues for holding sequential data and random-accessed data. In [16], the researches evaluate the four sequential prefetching algorithms. They use the threshold value for reducing the two non-beneficial effects from sequential prefetch including cache pollution and prefetch wastage. In compiler-aided approaches, the compiler makes the information for prefetching, then the helper thread organizes the prefetch based on compiler-generated information. [60] suggests the hybrid prefetch schemes between software prefetch schemes and hardware. The compiler generates the five classes of hints that the prefetch engine used for the prefetch. In [61], the compiler builds the acyclic blocks for the prefetch where the embedded information is used for handling pointer-intensive application. [17] provides the compiler that generates the pre-computing slice (p-slice) for handling inter-threaded dependencies. After generating the p-slice, it selects the *spawning pair* based on the dependency of p-slice for increasing the parallelism.

The dynamic schemes provides the adaptiveness at run time using hardware monitoring or preprocessing information. [62] optimizes the helper thread on CMP environments. They assign the helper threads on the idle processor and control the threads including the main thread and the helper thread using the shared memory. In [63], they provide the dynamic scheme to optimize the prefetch and control the helper thread based on the hardware information [64] minimizes the overhead of dynamic prefetching scheme using the co-location between the helper thread and the main thread. [23, 18] provide the optimal prefetch scheme for multi-level cache environments through optimizing low-level prefetching with the awareness of upper-level prefetching. [65] provides the Karma that supports the optimal cache solution by minimizing the overall disk access time.

B. Motivation

Many studies [42, 45, 7, 20, 44, 46, 51] have been carried out to design effective web prefetch mechanisms. However, existing prefetch schemes fail to provide a solution that is suitable for modern web frameworks. They do not perform well in conjunction with persistent HTTP, web cache schemes or request distribution schemes in a cluster environment.

1. HTTP 1.1 Framework

The HTTP 1.1. framework gives the small chance to prefetch embedded objects because of web pipeline. Generally, an HTML file contains a number of embedded objects. Web clients request the main HTML file to a server, then receive it. After analyzing the received main HTML file, web clients make the list of its embedded objects in the received main HTML file and request them to the web server. With HTTP 1.0, web clients establish new connection whenever it requests an object. A client with HTTP 1.1 reuses a connection for multiple requests using persistent HTTP. In Fig. 9, a web client with HTTP 1.0 has intervals between requests, while there is only one interval between the main object and its embedded objects with HTTP 1.1. These intervals between requests allow the web server to predict and prefetch the next requested web objects. In HTTP 1.0, there are intervals between consecutive requests that give the chance to prefetch the predicted embedded objects, while HTTP 1.1 makes it difficult to prefetch them. Web servers using HTTP 1.1, therefore, can prefetch embedded objects in the interval between the request to main object and the requests to embedded objects.

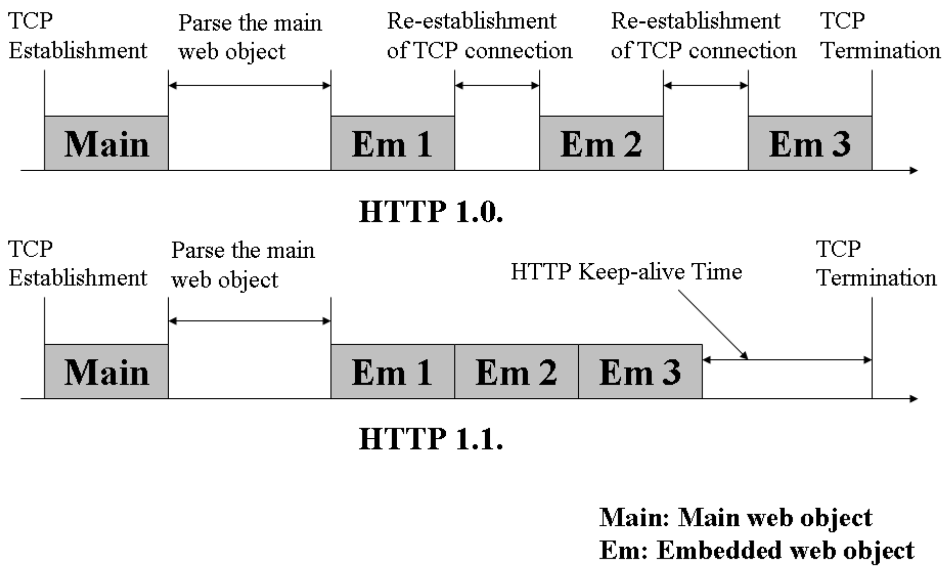

Fig. 9. Persistent HTTP and Web Pipeline

2. Web Cache Scheme in Cluster Environments

Incoming access pattern to the main object and the embedded objects in the web cluster server is random, while requests to the embedded objects follow request to the main object sequentially at clients or proxy servers. In general, web clients request the whole related embedded objects after processing the main object. However, we observe that only some embedded objects are requested after the main object. Table III shows the request frequencies of ten embedded objects after requesting 'main.html' in Computer Science and Engineering at Texas A&M University. Five embedded objects are frequently requested, while others have less than hundred requests.

One reason is that proxy servers provide some objects to the client directly if the requested objects are in the cache. The requests replied by the proxy server will not be transferred to the web server. In addition, the configuration of the web documents such as duration time and random objects can make the inequality of web access pattern. This random access pattern causes the misprediction. The misprediction does not only lose the chance to enhance the performance but also wastes the I/O and network bandwidth.

Table III. The Frequency of Request to the Embedded Objects

Index	Name	Request
1	/html4/front.css	517
2	/html4/global.css	517
3	/images/bin.jpg	477
4	/images/header.jpg	473
5	/images/LOOK.gif	446
6	/images/random/01	75
7	/images/random/06	69
8	/images/random/07	66
9	/images/random/09	66
10	/images/random/010	62

3. Web Prefetch Scheme in Cluster Environments

The distribution policy has influence on the effectiveness of the prefetch scheme in the web cluster system. The distribution policy in a web cluster system plays a crucial role in system performance [3, 29, 14]. Especially, the locality-based request distribution schemes [3, 29] enhance the performance of the web cluster systems through the efficient memory management of backend servers. When a client establishes one connection, prefetch schemes can be exploited in locality based distribution scheme. However, admission control for multiple connections per client, which is a new feature of HTTP 1.1, increases the difficulty of prefetching in web cluster environments. A web cluster handles multiple connections from the same client independently. Therefore, different backend servers in the web cluster reply to the same client. When a backend server prefetches the predicted objects, it is not guaranteed that the next incoming request will be forwarded to the backend server which has the web objects through prefetch. To handle multiple connections, the distributor should keep the prefetch information in the backend server of the web cluster system.

Moreover, the locality-based schemes with prefetching fail to achieve load balancing. The more files are prefetched at the backend servers, the more requests are forwarded to them. It results in consuming available memory space in the backend servers, and degrading the performance of the web cluster systems. A simply aggressive prefetch scheme may increase the inequality of request distribution and drop performance in web the cluster systems dramatically.

C. Adaptive Web Prefetch Scheme

To overcome the difficulty of the prefetch scheme at web cluster system, it is required the adaptive management scheme in web cluster environments. Prefetch schemes are composed of prediction of incoming requests and rate control of prefetch. We propose the adaptive prediction and rate control in web cluster systems using double P.P.M. scheme (*DPS*) and adaptive rate controller (*ARC*).

1. Double P.P.M. Scheme (*DPS*)

Although the previous works [57, 56, 55] provide hybrid schemes which lie between short-term and long-term approaches, they do not consider the modern web frameworks and cluster environments. At the server's side, access patterns are dynamic depending on the location of web clients and configuration of objects. The prefetch scheme at the server's side must be able to tolerate the randomness of access patterns.

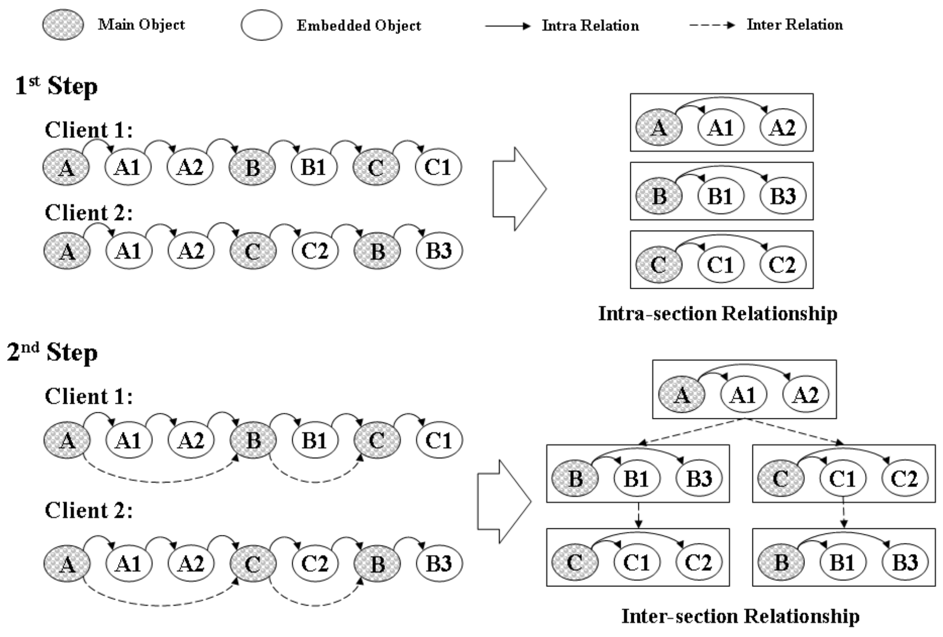

Fig. 10. Double P.P.M. Scheme

We propose a two-level hybrid scheme, referred to as *DPS*. *DPS* handles the main and embedded objects in different ways. Fig. 10 shows how *DPS* finds the relationship between web objects from web log files. For the first step, *DPS* classifies web objects into a number of groups each of which consists of one main object and its related embedded objects. During the grouping of web objects, *DPS* makes the record of the intra-section relationship defined by the request frequency of embedded objects after the request of the main object. *DPS* differentiates a the main object from the embedded objects according to the file extensions. Web objects with file extension such as 'html', 'php' and 'jsp' are classified into main objects. In Fig. 10, gray circles, white circles and arrows denote main objects, embedded objects and relationships, respectively. Each arrow has a prediction value which shows the probability that the request of an embedded object will follow that of a main object. In Fig. 10, *DPS* finds three groups including 'A', 'B' and 'C'. In the second step, *DPS* searches for the inter-section relationship between groups. *DPS* focuses on only the access to main objects that is representative of the group. In this step, *DPS* defines the access to main objects as the access to their corresponding groups. In Fig. 10, *DPS* detects two access patterns to main objects including 'A $\rightarrow$ B $\rightarrow$ C' and 'A $\rightarrow$ C $\rightarrow$ B'. Although access patterns are changed by the web cache or configuration of web objects, *DPS* can find access patterns of objects accurately.

2. Adaptive Rate Controller (*ARC*)

Using the relationship information provided by *DPS*, Adaptive Rate Controller (*ARC*) calculates the prefetch rate dynamically. It determines which objects should be prefetched considering memory status. Memory is one of the critical resources in web server system. Aggressive prefetch schemes does not always guarantee the performance enhancement.

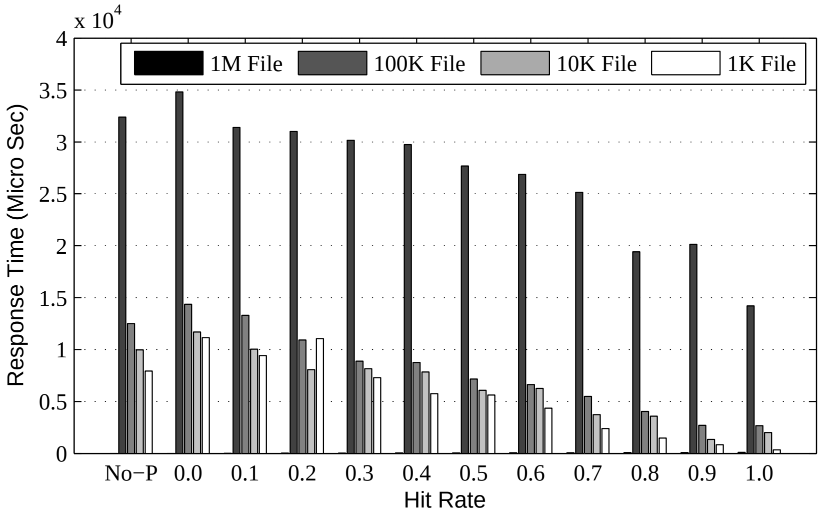

Fig. 11. Prefetch Efficiency

Fig. 11 shows the response time over the variable hit rate of the prefetched data. We use 100 file groups for each file size; 1MB, 100KB, 10KB and 1KB. A file group includes 20 files of the same size. A client selects one group, and then generates requests to two files in the group at an interval of 2 seconds. After processing the first incoming request, the web server prefetches one file from the disk. The client creates the requests to the prefetched object or non-prefetched object according to the test configurations. With less than 10% hit rates, it takes longer than non-prefetch scheme. This result proves that the inefficient memory management degrades the performance in the web server system. It is a critical problem to figure out which files are prefetched in the memory.

We should formulate system improvements considering disk workloads and memory status. The following equation shows the access improvement.

$$P = RT - RT^{'}, \tag{3.1}$$

where RT and $RT^{'}$ are the average response time without prefetching the predicted object and the average response time with prefetching the predicted object, respectively. ARC manages the prefetch memory for P in equation (3.1) to be positive. A web server is modeled by employing an M/G/1 round-robin queuing system where web server processes share one CPU and disk. Also, all cached pages in buffer cache have the same access probability in the future. First, the average response time (RT) in a web system is

$$RT = H_{sys} \times T_{mem} + (1 - H_{sys}) \times T_{disk}, \tag{3.2}$$

where H_{sys}, T_{mem} and T_{disk} are the hit rate of memory, memory transmission and disk response time, respectively. T_{disk} is the disk response time denoted by $(DS)/(1-\rho)$, where DS is the disk service time and ρ is the disk utilization. Disk utilization, ρ, is the miss rate of memory multiplied by the disk service time, DS. The miss rate of memory is $\lambda \times (1 - H_{sys})$, where λ is the request rate to a page.

When requests hit on memory, requested pages are located in the prefetch memory or the buffer cache. Thus, H_{sys} is $H_{pref} + H_{buf}$, where H_{pref} is the hit rate of prefetch memory and H_{buf} is the hit rate of buffer cache. We drive equation (3.2) to equation (3.3) as follows.

$$RT = (H_{pref} + H_{buf}) \times T_{mem} + F_{sys} \times \frac{DS}{\Delta}, \tag{3.3}$$

where F_{sys} is the miss rate of memory, the same as $1 - H_{sys}$. Δ is $1 - \lambda \times F_{sys} \times DS$. H_{pref} and H_{buf} are the hit rates of prefetch memory and buffer cache, respectively. When the requested page misses on memory, the response time becomes $(DS)/\Delta$, which is greater than zero. Also, DS and Δ are greater than zero.

$$\Delta = 1 - \lambda \times F_{sys} \times DS > 0 \tag{3.4}$$

When there is not enough space to prefetch related files, operating systems remove some cached objects in buffer cache to increase available memory space. It increases the hit rate of prefetch memory, while decreasing the hit rate of buffer cache. After prefetching related files, the average response time $(RT^{'})$ is defined as below.

$$RT^{'} = (H_{pref}^{'} + H_{buf}^{'}) \times T_{mem} + F_{sys}^{'} \times \frac{DS}{\Delta^{'}} \tag{3.5}$$

where $H_{pref}^{'}$, $H_{buf}^{'}$ and $F_{sys}^{'}$ are the hit rate of prefetch memory, the hit rate of buffer cache and the miss rate of memory, respectively. $\Delta^{'}$ is $1 - \lambda^{'} \times F_{sys}^{'} \times DS$, where $\lambda^{'}$ is the request rate to a page. When the requested page misses, the response time becomes $(DS)/(1 - \lambda^{'} \times F_{sys}^{'} \times DS)$, which is greater than zero. Also, DS and $\Delta^{'}$ are greater than zero.

$$\Delta' = 1 - \lambda' \times {F_{sys}}' \times DS > 0 \tag{3.6}$$

The access time to cached objects in memory is negligible, therefore T_{mem} becomes zero. We rewrite equation (3.1) as below.

$$P = RT - RT' = \frac{DS \times (F_{sys} \times \Delta' - {F_{sys}}' \times \Delta)}{\Delta \times \Delta'} \tag{3.7}$$

For P in equation (3.7) to be positive, Δ, Δ', DS and $F_{sys} \times \Delta' - {F_{sys}}' \times \Delta$ should be also positive. In equations (3.4) and (3.6), Δ and Δ' are greater than zero. DS is positive, because it is the disk service time. Therefore, $F_{sys} \times \Delta' - {F_{sys}}' \times \Delta$ should be greater than zero. In equation (3.5), Δ' is $1 - \lambda' \times {F_{sys}}' \times DS$. $F_{sys} \times \Delta' - {F_{sys}}' \times \Delta > 0$ is rewritten as below.

$$\begin{aligned} F_{sys} &- \lambda' \times {F_{sys}}' \times F_{sys} \times DS - {F_{sys}}' \\ &+ \lambda \times {F_{sys}}' \times F_{sys} \times DS > 0 \end{aligned} \tag{3.8}$$

Web prefetch system processes not only demanded requests but also prefetch requests. Therefore, λ' is the sum of λ for demanded requests and λ_{pref} for prefetch requests. We rewrite inequality (3.8) to (3.9).

$${F_{sys}}' < \frac{F_{sys}}{1 + \lambda_{pref} \times F_{sys} \times DS} \tag{3.9}$$

${F_{sys}}'$ is $1 - {H_{pref}}' - {H_{buf}}'$, where ${H_{pref}}'$ and ${H_{buf}}'$ are the hit rates of prefetch memory and buffer cache, respectively. We assume the cached objects have the same probability to be accessed in the future. Therefore, all pages in buffer cache contribute uniformly to the H_{buf}. When $N(B)$ pages are in buffer cache, each page contributes $(H_{buf})/(N(B))$ to the cache hit ratio. When more objects are prefetched, ${H_{pref}}'$ increases. ${H_{pref}}'$ is the sum of H_{pref} and H_{pred} where H_{pred} is the prediction value of prefetched files.

$$\begin{aligned} H_{pred} > 1 \quad &- \quad H_{pref} - H_{buf} \times \frac{N(B) - N(R)}{N(B)} \\ &- \quad \frac{F_{sys}}{1 + \lambda_{pref} \times F_{sys} \times DS}, \end{aligned} \tag{3.10}$$

where $N(B)$ and $N(R)$ are the buffer cache size and the released buffer size for prefetched object, respectively. To maximize the performance improvement, our web predictor monitors the hit rate and size of the buffer cache, and then prefetches web objects which satisfy inequality (3.10). We get prediction values from *DPS*, while the hit rate and size of buffer cache are obtained using PAPI [66] and 'meminfo' in Linux Kernel.

D. Prefetch Memory Management

For management of prefetch memory, our prefetch scheme has two hash tables; one table obtains the information for the transition graph generated by *DPS* and the other table includes the information for the prefetched file on the user memory. The transition table helps the prefetch process to predict the future requests. In Fig. 12, the prefetch table has the prediction values that show access probability to the prefetched objects. There are prefetch process in the backend nodes that manages to prefetch the predicted objects.

A web server has two kinds of processes; web server processes and web prefetch process. Web processes handle the incoming requests and web prefetch process supports the prefetch scheme for the next incoming requests. In the memory of backend servers, there are four space groups including system area, web server process area, prefetch memory area and buffer cache area. System area is used for system management processes such as kernel, network management processes, I/O management

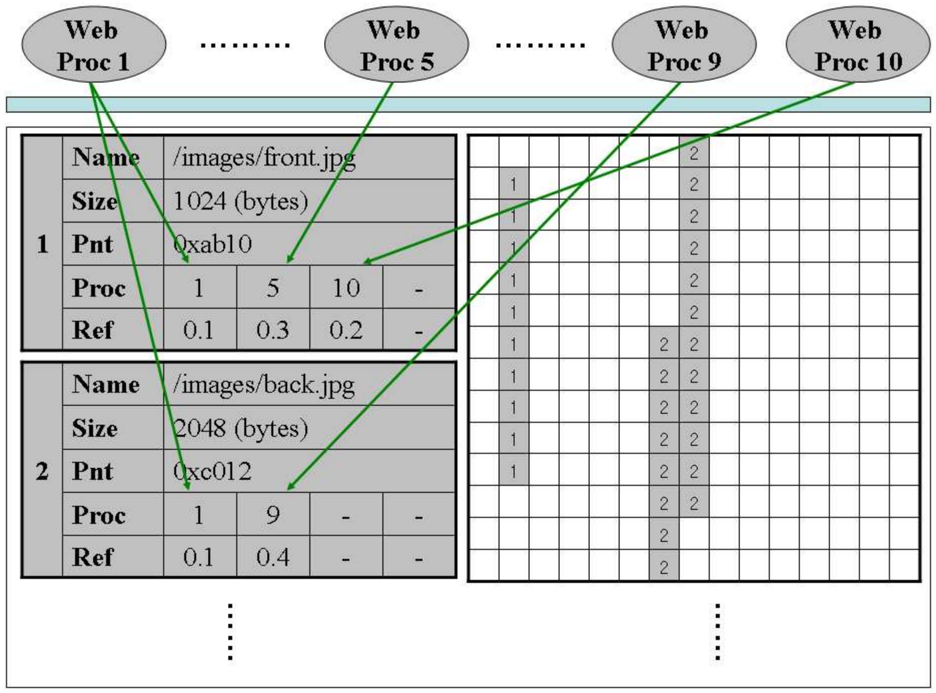

Fig. 12. Web Prefetch Memory Management

processes and security management processes. Web server process area is for running web server processes. Prefetch memory area holds the prefetched objects and prefetch information, and buffer cache area keeps files for the improvement of the performance of I/O. Our *ARC* increases the efficiency of I/O.

1. Insertion of Web Objects to Prefetch Memory

After replying the requested file to the client, the prefetch process predicts the file which can be requested in the future. After making the list of the predicted files, the prefetch process checks whether the file is already prefetched or not in the prefetch table. If it is already prefetched, the prediction value of the prefetched object is increased. If not prefetched, the prefetch process reads from the buffer cache area or the disk and stores them it in the prefetch memory. After prefetching the predicted file, the prefetch process reports the list of the prefetched files to the distributor. In Fig. 12, the web process 1 prefetches two web objects including 'front.jpg' and 'back.jpg', while the web process 5 prefetched only the file 'front.jpg'.

2. Deletion of Web Objects from Prefetch Memory

After terminating the web connection to access, the prediction value of the prefetch file is decreased. If the prediction value is 0, the prefetched file is deleted from the prefetch memory. In Fig. 12, if connections with the 'web proc 1' and the 'web proc 9' are terminated, 'back.jpg' is deleted from the prefetched memory.

3. Retrieval of Web Objects from Prefetch Memory

When a new request is incoming, the web predictor checks whether the requested object is already prefetched or not. If the requested web object is prefetched, the prefetch process gives the access point of the prefetched file to the web server. Then, the web server processes the incoming request without disk access. If not, the web server handles incoming requests as normal.

E. Performance Evaluation

1. System Configuration

Our simulator is composed of two days of web traces, a web analyzer and a web cluster simulator. The web analyzer obtains the relation information based on the first day's web traces. Then, the web cluster simulator simulates prefetch schemes using the second day's web traces and the information from the web analyzer.

Fig. 13 shows the detailed kernel components related with file I/O operation. We design the memory management module in the web cluster system running on the

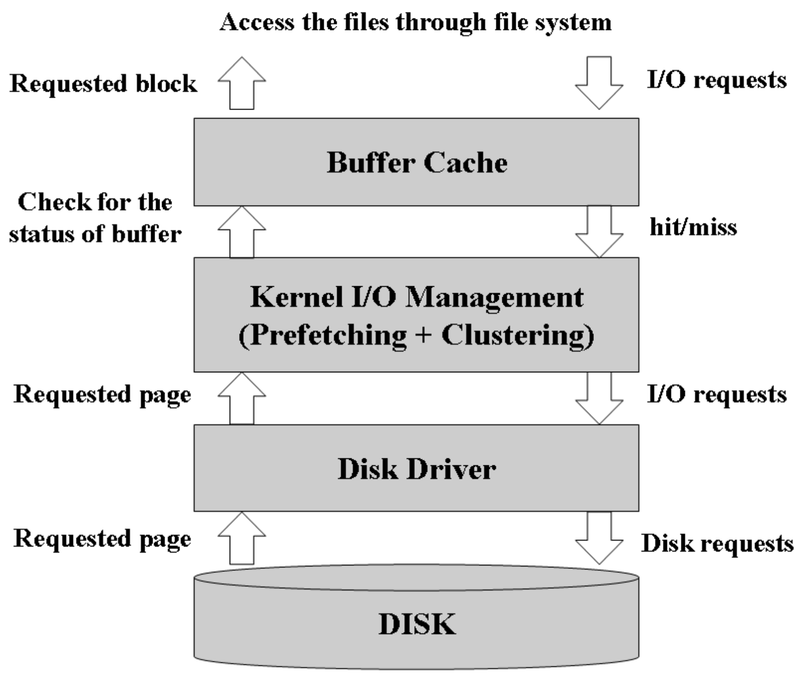

Fig. 13. Kernel Components in Disk I/O Operation

Linux kernel 2.6. When the web server processes access web objects including files through the kernel, kernel components work together to improve I/O performance. The buffer cache releases the overhead of disks reducing the number of I/O requests to disks. The I/O prefetch module reads consecutive blocks in advance, while I/O cluster module reads a cluster of blocks at a time. We design I/O operation in our simulator based on the Linux kernel's I/O prefetch and cluster schemes. The least frequency used (LRU) scheme is employed as a cache replacement policy in our simulation. The web cluster system is configured with one web distributor and four backend servers. Each backend server has its own disk that contains the whole web objects. In addition, the disk simulator, DiskSim [34], is embedded in our simulator for accurate low-level I/O simulation. The simulation parameters are shown in Table IV.

Table IV. Simulation Parameters (Web Prefetch Scheme)

Backend Server	
Parameter	Value
Physical Memory	512 MB
Kernel Space	100 MB
Web Process Size	20 MB
Disk Latency	Provided by DiskSim [34]
Memory Latency (Hit)	500 Mbps
Network Latency	100 Mbps
Web Distributor	
Splice Hit	90.44 in Table I
Splice Miss	2362.927 in Table I
Session Time	15 Seconds [35]
Web Client	
Concurrent Connection	Up to 4

We measure the size of the kernel memory and web server processes while running Apache 2.2 on the Linux Kernel 2.6.18. Each web client can create up to four connections for delivering web documents, and each backend server maintains 5 to 10 idle web processes for future requests.

To measure the duration time of distributing requests, we install the simple client and modified distributor at a machine. Computer has Intel Core 2 duo processor and two GB memory. The local area network operates at the speed of 100 Mbps. A simple web client sends the requests to the distributor, then the web switch forwards the incoming requests into backend server. After receiving it, the backend server replies the dummy packet into the web switch. Finally, the distributor relays the packets from web server into simple web clients. The simple web clients measure the elapsed time during the request processing. We test 1000 requests using the implemented machine at hit/miss cases.

In the simulation, the web distributor process the incoming requests using the locality-based request distribution. Too much web connections in persistent HTTP consume the whole available memory and even use the swap area. The web distributor controls the maximum number of connections to avoid the memory saturation in the backend servers.

Table V. Requests to the Embedded Objects

Name	Day 1	Day 2	HTTP
CS TAMU	25479	20018	HTTP 1.1
NASA	64714	60265	HTTP 1.0
ClarkNet	210908	229944	HTTP 1.0
SPECweb2005-1	6304	116302	HTTP 1.1
SPECweb2005-2	12018	116045	HTTP 1.1

Table V shows three real traces from web sites including Department of Computer Science and Engineering in Texas A&M University, NASA and ClarkNet. Also, we use two synthetic traces from SPECweb2005 benchmark [67]. Although web clients use HTTP 1.0 in the ClarkNet and NASA workloads, we assume that they use the persistent HTTP.

2. Evaluation Results

For evaluating the performance of our adaptive schemes, we compare two scenarios including distribution without prefetch scheme and various prefetch schemes. First, we compare the performance of distribution schemes including round robin, LARD, persistent LARD and *ProRD*. We select the most efficient distribution scheme and apply it to various prefetch schemes. Second, we compare the performance of our adaptive prefetch scheme to the existing P.P.M. schemes [43, 40] and cluster schemes [45, 46]. Our adaptive scheme uses *DPS* to get the relation information of web objects and *ARC* to dynamically calculate the threshold value for prefetching. P.P.M. and cluster schemes use six different static threshold values including '0.0', '0.2', '0.4'. '0.6', '0.8' and '1.0'. For the completeness of our study, we also include results of non-prefetching scheme.

a. Request Distribution

Fig. 14 shows request distribution in backend nodes under ClarkNet workload. X axis represents prefetch schemes and Y axis denotes standard deviation of incoming requests. First 12 entries in X axis show the static schemes using P.P.M and cluster schemes. NO_P and *ADA* denote non-prefetching and our adaptive prefetch schemes, respectively. We run the simulations in three different orders for each prediction scheme. The first order scheme predict the future requests based on only one previous request, while the second order scheme and the third order scheme uses two and three requests for the future request predictions. When web prefetch schemes cannot find future requests at high-order prediction, they use low-order prediction. In the case of aggressive prefetch schemes, they try to prefetch more objects in some of backend nodes, which results in a skewed request distribution. However, *ADA* distributes requests almost as fairly as a non-prefetch scheme. We use the *ProRD* distribution scheme at the overall prefetch schemes in the simulation. Fig. 14 shows the result of the simulation for TAMU and ClarkNet workloads. The results of other workloads show in Appendix A.

Fig. 15 shows how many requests are not provided by the web cluster schemes. X

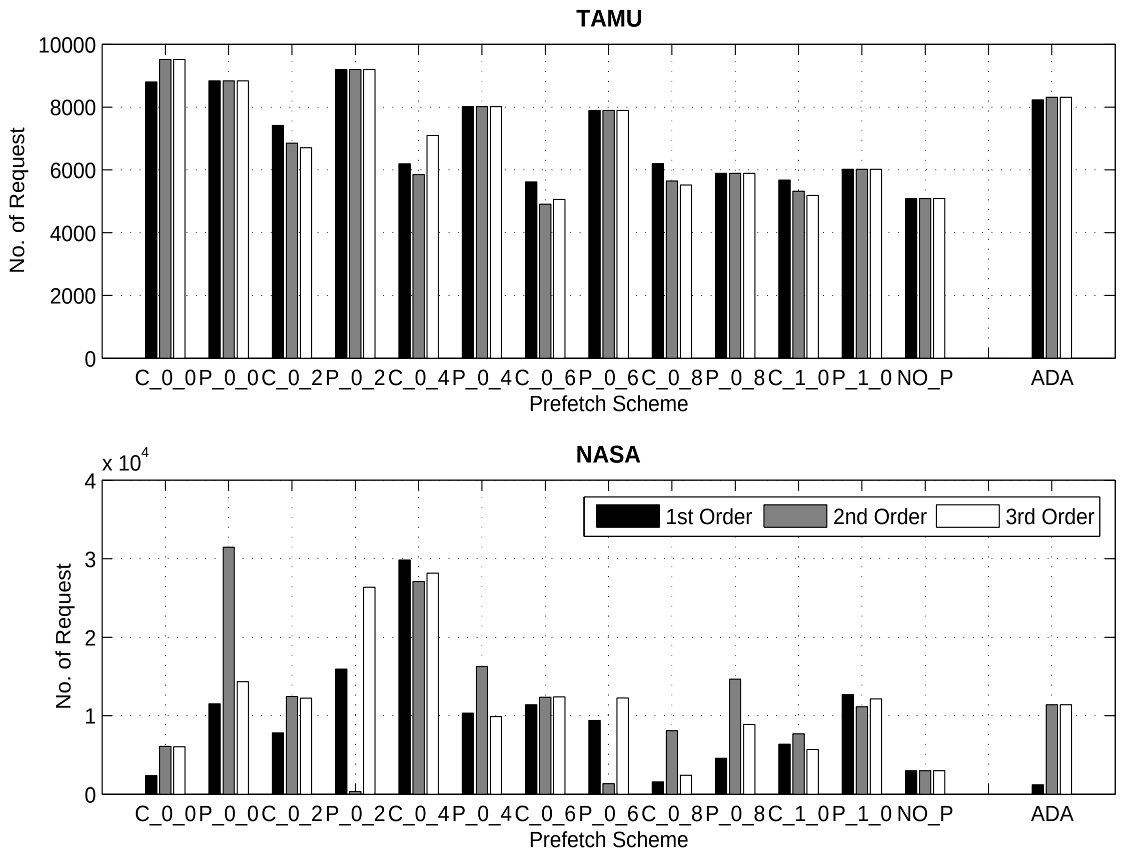

Fig. 14. Standard Deviation of Distribution (TAMU, NASA)

axis in the graph shows the prediction schemes in the simulation, while Y axis means the number of the missed requests in ClarkNet workload. At other workloads, web cluster systems can handle the whole request. However, the ClarkNet workload incurs the missed requests because of the shortage of the memory. Basically, backend servers limits the number of web processes and connections, because too many web processes consume the whole memory and incur the swap operation. In Fig. 14, the aggressive schemes provide the skewed distributions of incoming requests. It multiplies memory consumption to handle the web client and prefetched objects. Therefore, memory shortage causes the missed requests. In the 0.0 threshold, the cluster scheme has more missed requests than P.P.M. schemes, while the P.P.M. schemes lose more requests

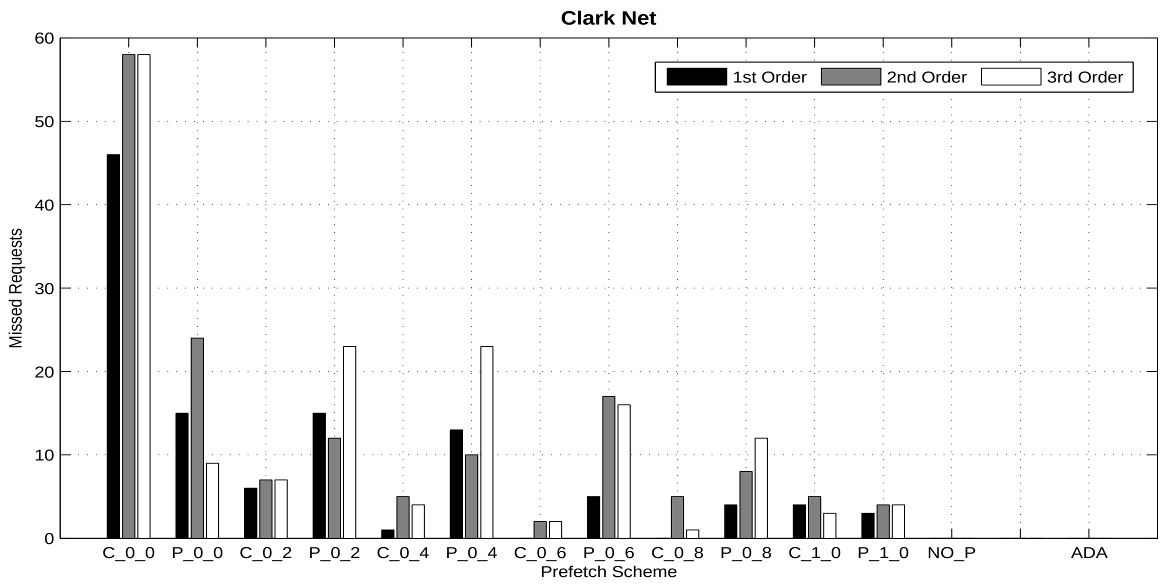

Fig. 15. Missed Requests

than the cluster schemes at the 0.2, 0.4 and 0.8 thresholds. Under the 0.2, 0.4 and 0.8 thresholds, the P.P.M. schemes prefetch more objects than the cluster schemes in the figure on page 56. P.P.M. schemes consume more memory to handle the prefetched objects and increases the probability to miss the request.

b. Web Prefetch Schemes

In Fig. 16, Y axis denotes the response time from web cluster system and X axis represents prefetch schemes. First 12 entries in X axis show the static schemes using P.P.M and cluster schemes. NO_P and *ADA* denote non-prefetching and our adaptive prefetch schemes, respectively. We run the simulations in three different orders for each prediction scheme. The first order scheme predict the future requests based on only one previous request, while the second order scheme and the third order scheme uses two and three requests for the future request predictions. When web prefetch schemes cannot find future requests at high-order prediction, they use low-order prediction.

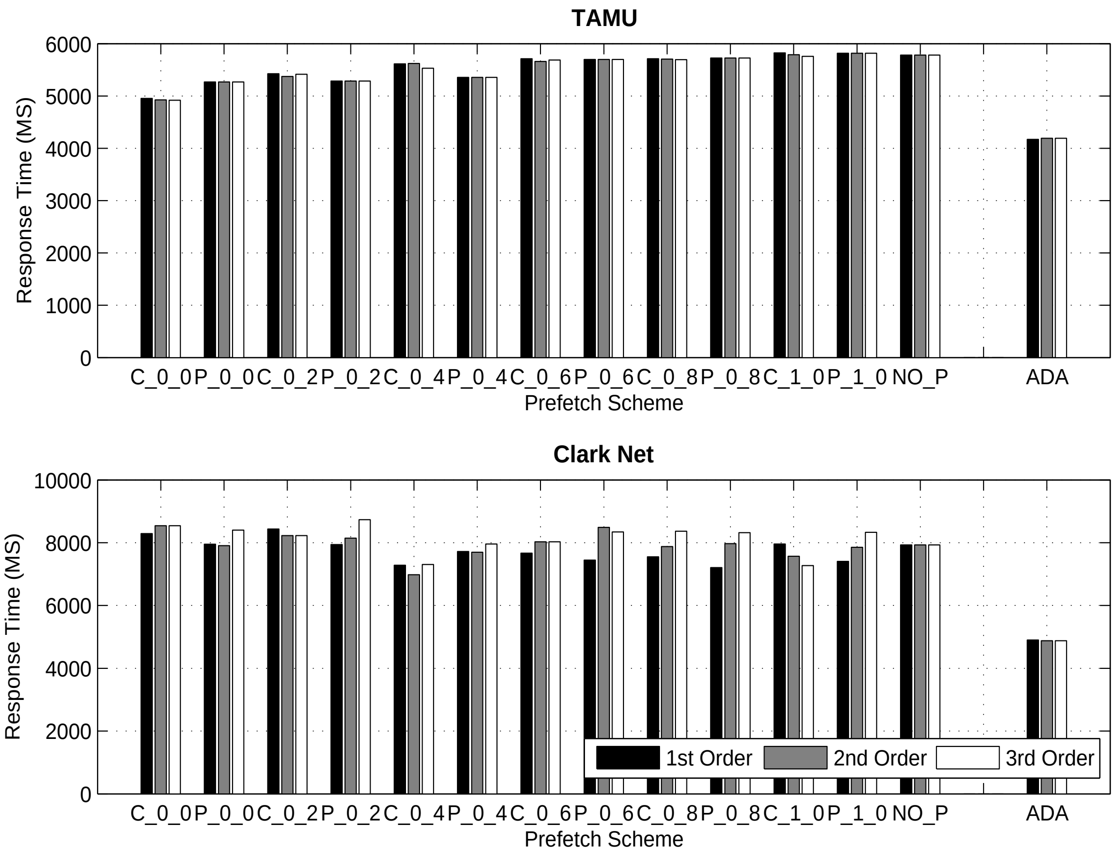

Fig. 16. Web Response Time in Prefetch Schemes (TAMU, ClarkNet)

In Fig. 16, *ADA* scheme outperforms others by maximum 40 percent. In light web workloads such as TAMU and NASA, aggressive web prefetch schemes reduce the response time. They move web objects on the memory to enhance the hit rate and reduce the access to the disk. However, in heavy workloads such as ClarkNet, aggressive prefetch schemes increase the response time. The prefetch scheme with 0.4 threshold shows the best response time among the static schemes. It shows that aggressive prefetch schemes increase the overheads of the web cluster system. Although a high-order prediction scheme gives more accurate prediction of future requests, the performance gap between high and low order is not distinguishable. High-order schemes even show worse performance than low-order schemes, as in threshold 1 of TAMU or threshold 0.8 and 1 of ClarkNet. Cluster schemes prefetch more web objects at low threshold than P.P.M. schemes. This makes cluster schemes show the fast response time with high hit on prefetch memory. With high threshold, cluster schemes do not find the web objects over the threshold. Therefore, P.P.M. schemes show similar to or better performance than cluster schemes at a high threshold. Fig. 16 shows the result of simulation for TAMU and ClarkNet, while other results show Appendix A.

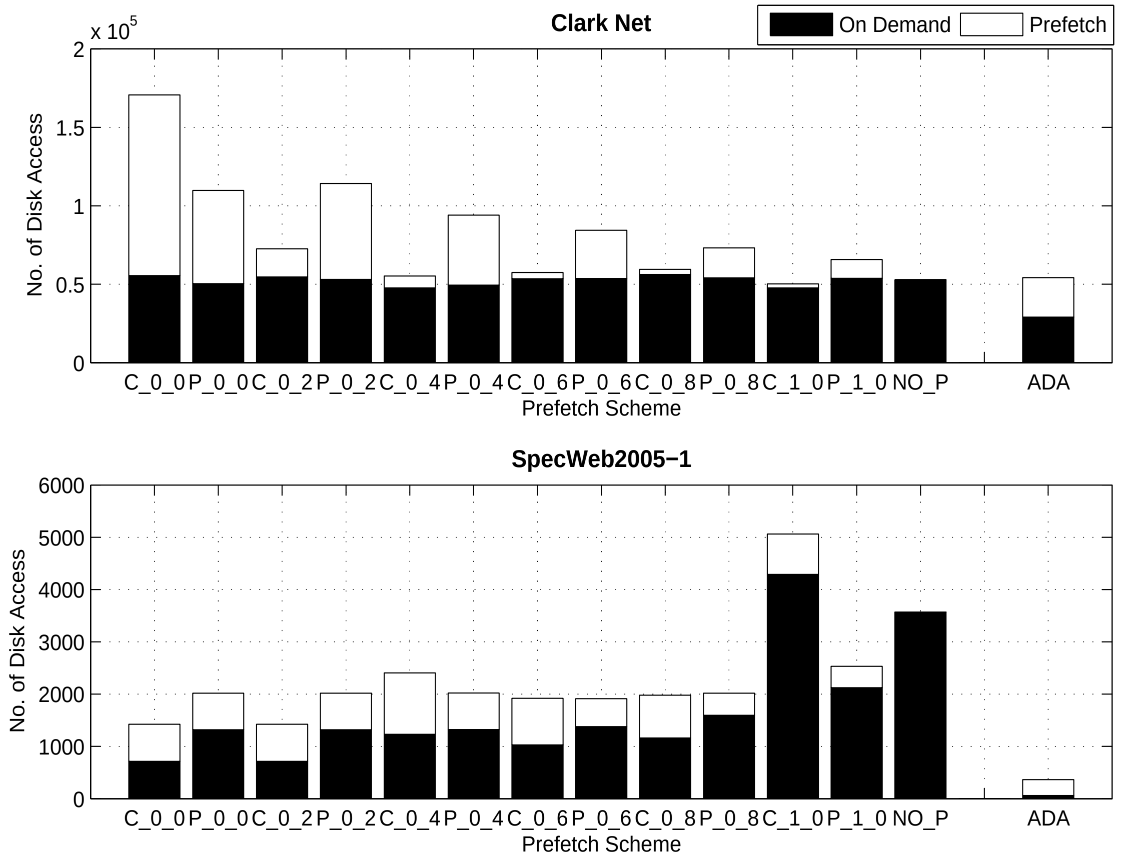

Fig. 17. Number of Disk Access (ClarkNet, SpecWeb2005-1)

Fig. 17 shows the number of disk access of web servers to retrieve web objects. The black bar shows the accesses to the requested web objects, while white bars means the number of accesses to web objects for prefetch. 'No_P' schemes do not have the white sections, because it does not prefetch. In the simulation of TAMU workload, prefetch scheme with low threshold has more disk access than prefetch scheme with high threshold. The prefetch scheme with low threshold do frequent disk access to prefetch web objects. Even though they incur the overheads on the storage, the prefetch schemes with low threshold reduce the disk access to the requested web objects. In the NASA workloads, the client uses HTTP 1.0 that has the interval between requests. P.P.M. schemes predict future requests during each interval, while cluster schemes handles the bundle of requests only when main requests are incoming. P.P.M. schemes provide more accurate prediction and reduces the frequency of disk access on the demanding time. In the ClarkNet workloads, prefetch schemes with low threshold does not reduce the disk access, and frequent disk accesses increase the response time due to high overheads. In the SpecWeb2005 workloads, clients establish HTTP connections using HTTP 1.1 that reuses the TCP connection for HTTP requests. Cluster schemes provide more accurate prediction than P.P.M. scheme. 'C_1_0' cluster scheme in SpecWeb2005-1 workload removes the cached web objects on the memory that will be accessed in the future. Fig. 17 shows the result of the simulation for ClarkNet and SpecWeb2005-1 workloads. The results of other workloads show in Appendix A.

Fig. 18 shows the number of the hit on the prefetch memory and the buffer cache. The prefetch memory is allocated on the user memory, while buffer cache is allocated on the kernel memory. The numbers of hit on the kernel memory is reduced when the number of hits on prefetch memory are increased. If memory has available space, the prefetch scheme loads the objects without removing objects in the memory. When the whole memory is occupied, it takes the available memory from releasing the buffer cache. When web objects are prefetched, they reduce the probability of the hit on the

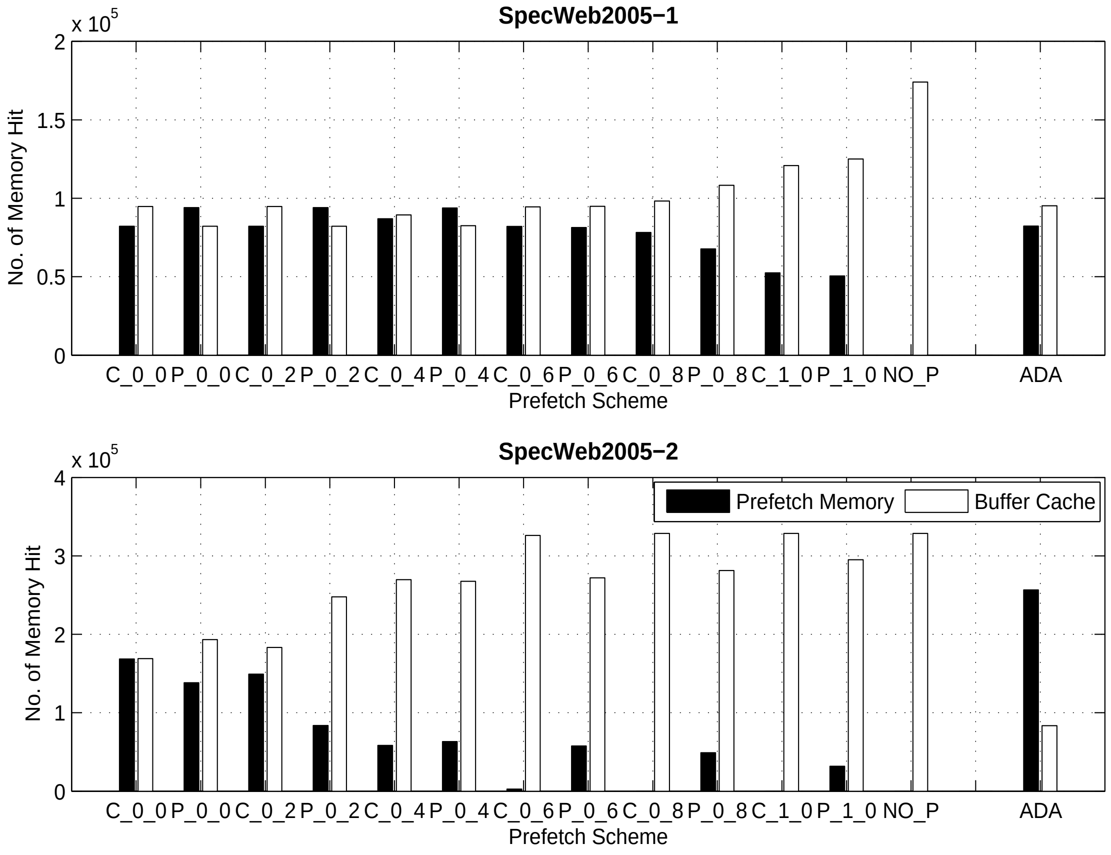

Fig. 18. Number of Hits on Memory (SpecWeb2005-1, SpecWeb2005-2)

kernel memory. The *ADA* prefetch scheme under ClarkNet workload show the high numbers of hit on the prefetch memory and buffer cache at same time. The *ADA* controls the prefetch rate in order to avoid a drop of efficiency of the buffer cache. The 'C_0_0' prefetch scheme in ClarkNet workload increases the hit rate of prefetch memory, but reduces the hit rate of buffer cache. Therefore, the 'C_0_0' prefetch scheme shows longer response time than the 'C_0_4' prefetch scheme in Fig. 16. When a web cluster runs on the TAMU and NASA workloads, the prefetch scheme with low threshold shows the high numbers of hit on the prefetch memory. Because few web objects strongly related with the requested web objects, the prefetch scheme with high threshold loads small number of web objects into the memory. SpecWeb2005-1 workloads show the high relationship between web objects. The numbers of hit on prefetch memory are higher than other schemes. The *ADA* shows the high numbers of hit on the prefetch memory. But, it is not the highest one in prefetch scheme. The 'P_0_0' and the 'P_0_2' prefetch schemes are higher that other schemes, but it reduces the numbers of hit on buffer cache. In the SpecWeb2005-2 workloads, the numbers of hit on the memory of the *ADA* scheme shows the different pattern where the numbers of hit on prefetch memory is higher than aggressive prefetch schemes. The *ADA* scheme manages the prefetch memory efficiently. Fig. 18 shows the result of the simulation for SpecWeb2005-1 and SpecWeb2005-2 workloads. The results of other workloads show in Appendix A.

Fig. 19 shows the average access time to the disk when web processes access the requested web objects. Web processes has zero disk access time when the requested file in the memory. At the miss of the requested objects, web processes take the suitable time to get object from disk. The heavy disk workloads such as ClarkNet and SpecWeb2005-2 incur the long disk access time. The TAMU workload and the SpecWeb2005-1 workload can access the web objects in the disk in a short time. In the TAMU and NASA workloads, aggressive schemes reduce the disk access time, because it increases the hit of the requested web objects. However, aggressive schemes in the ClakrNet workload shows the long disk access time. Aggressive schemes in ClarkNet workload have the similar numbers of hit as other schemes, but it accesses disk frequently for prefetching web objects. Therefore, it increases the average access time to the disk. In the SpecWeb2005-1 workload, the 'C_1_0' scheme has the frequent access to the disk at the demanding time in Fig. 18. As a result, the 'C_1_0' scheme shows longer disk access time than even 'NO_P' scheme. In the SpecWeb2005-2 workloads, 'C_0_6', 'C_0_8' and 'C_1_0' schemes have no hit on the prefetched memory and access to disk. Therefore, the disk access times in 'C_0_6', 'C_0_8' and 'C_1_0'

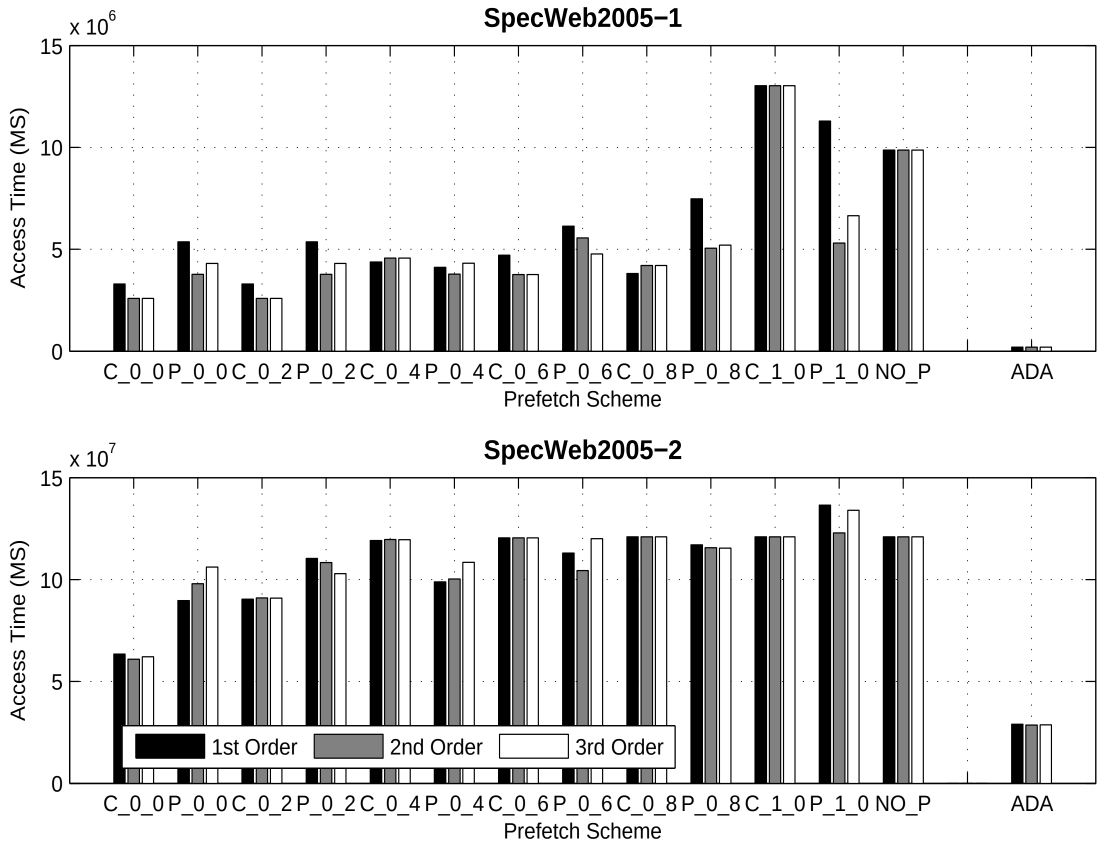

Fig. 19. Disk Access Time on the Demanding Requests (SpecWeb2005-1, SpecWeb2005-2)

schemes are similar. Fig. 19 shows the result of the simulation for SpecWeb2005-1 and SpecWeb2005-2 workloads. The results of other workloads show in Appendix A.

c. Usage of Prefetch Memory

Fig. 20 shows how many objects are prefetched under various prefetch schemes. X axis is the prefetch scheme in the simulation, while Y axis shows the number of the prefetched block. When web processes access the file in the disk, they read it by 256 bytes. Aggressive schemes prefetch more web obejcts than other schemes. The '*ADA*' scheme prefetches more web objects than even 'C_0_0' schemes in SpecWeb2005-1 and

NASA workloads. The '*ADA*' scheme distributes avoids the skewed distribution in Fig. 14. But the 'C_0_0' scheme keeps forwarding requests into the same backend

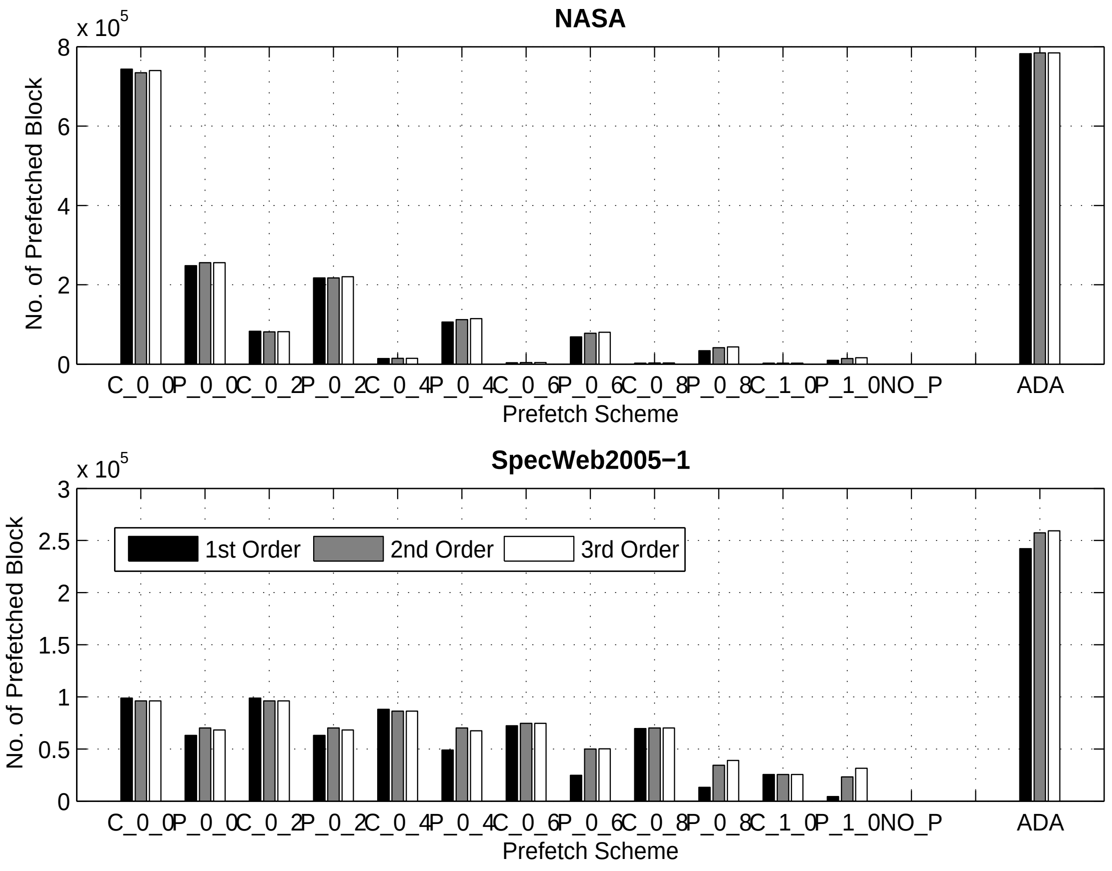

Fig. 20. Prefetched Blocks (NASA, SpecWeb2005-1)

server for increasing the numbers of hit on the prefetch memory. In a cluster scheme, one cluster has lots of web objects at a low threshold, while the number of objects in a cluster is decreased at a high threshold. At the high threshold prediction scheme, the P.P.M. schemes prefetches more web objects than cluster schemes. Other results except SpecWeb2005-1 and NASA workloads in Fig. 20 show in Appendix A.

Fig. 21 shows the usage of the prefetched block. X axis represents the prefetch schemes in the simulation, while Y axis shows that the accesses of the prefetched block are divided by the number of prefetched block. Even though aggressive schemes show

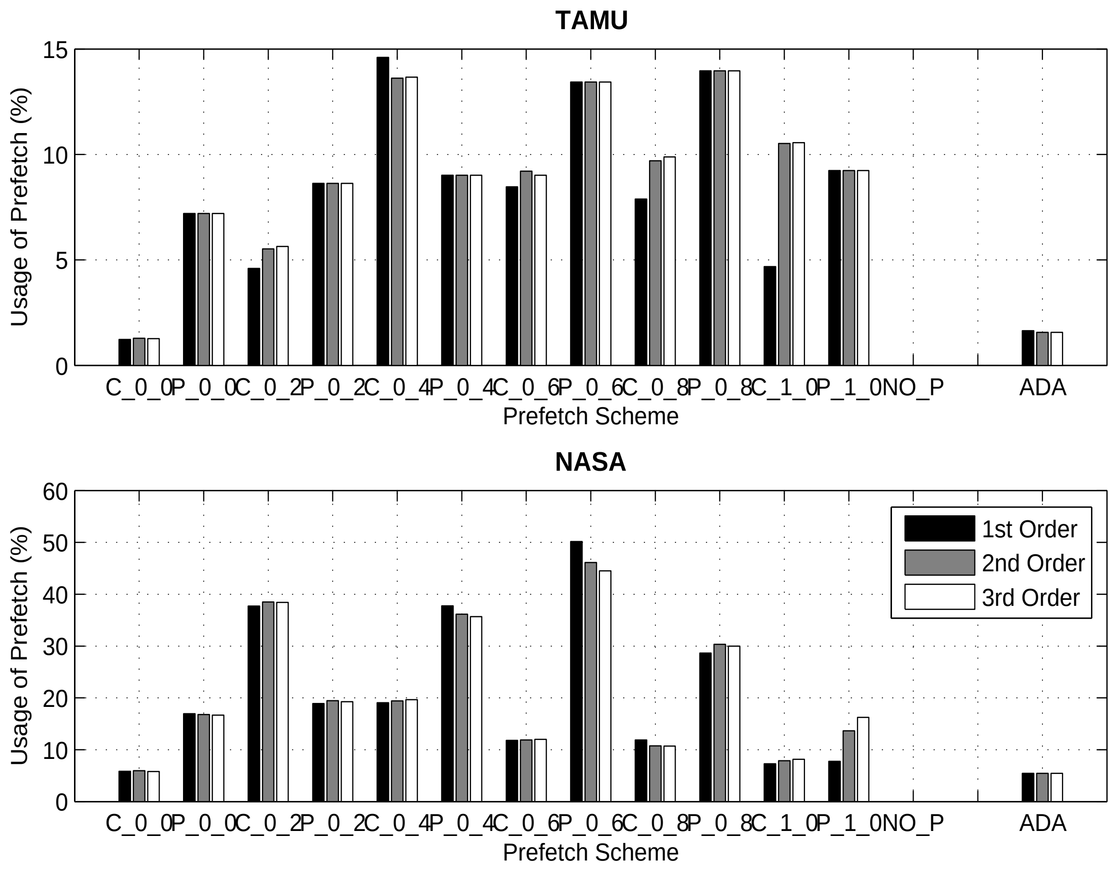

Fig. 21. Usage of Prefetched Blocks (TAMU, NASA)

the high numbers of hit on the prefetch memory, it prefetches the block that is not accessed at the future requests. The *ADA* scheme has a low usage of prefetched blocks. The *ADA* scheme in the ClarkNet workload has a low efficiency on prefetched block. Even though 'C_0_0' scheme prefetches more web objects and shows higher numbers of hit on prefetch memory in Fig. 18 and Fig. 20, it increases the overhead on the I/O system and low numbers of hit on the buffer cache in Fig. 18 and Fig. 19.

Prediction schemes with the 1.0 threshold shows the lower usage than prediction schemes with lower thresholds. In the TAMU workload, 'C_0_0' schemes shows the highest usage. 'P_0_6' scheme has the highest usage of prefetched block in the NASA workload. The frequency of the web objects can make the misprediction in each prediction scheme. For example, when requests to object A are logged at two times, requests to object B are incoming at two times following by requests to A and request to object C is incoming at only one time. The relationship between object A and object B is 1.0, while the relationship between A and C is 0.5. But it can also happen accidentally. This misprediction decreases the usage of prefetched object in the prediction scheme with high threshold. In SpecWeb2005-1 and SpecWeb2005-2 workloads, some prefetched schemes show over 100 percent usage on prefetched object. Web preocesses can access the objects that are already prefetched by other web processes. It can increases the usage by over 100 percent in prediction scheme. Fig. 21 shows the result of simulation for TAMU and ClarkNet, while other results show Appendix A.

DPS has a great advantage in modern web framework compared with P.P.M and cluster schemes. It can tolerate the randomness of access patterns caused by web caches or clients. *ARC* monitors the hit rate and the size of the buffer cache and calculates the optimal prefetch threshold dynamically. It makes a decision of prefetching using the dynamic threshold and relationship information provided by *DPS*. Since the optimal prefetch rate keeps changing over time depending on the system condition, using the static threshold values is not a good solution to manage the system properly. A prefetch rate that is too high drops the performance through the shortage of memory and I/O bandwidth, and a prefetch rate that is too low loses the chance to improve the performance through prefetching.

CHAPTER IV

ADAPTIVE STREAMING SERVICE

A. Background

Multimedia streaming service consumes lots of bandwidth in the network systems. For the management of multimedia streaming, many previous studies retrieve information from network and system. Based on the information, clients or intermediate node handle multimedia streaming efficiently. In the streaming service, bandwidth estimation decide the quality of streams and provide the efficient stream. This chapter shows the previous work for estimating the available bandwidth.

1. Bandwidth Estimation in Wired Network

Since the introduction of Cprobe [8], a method for estimating bandwidth using Internet Control Message Protocol (ICMP) packet trains, many tools have been suggested. The basic scheme in the available bandwidth estimation used a statistical analysis of the received probe packets in the destination node. The existing schemes are classified into two groups; probe gap [11, 10, 68] and probe rates [9, 10, 69].

In probe rate schemes, the comparison of the incoming rate from the sender side to the outgoing rate reveals the incoming rate to be less than or equal to the available bandwidth of the probing link: Pathload, PTR and TOPP.

In the Pathload [27], the scheme can measure the available bandwidth and capacity at a same time. They define the available bandwidth as the maximum throughput that can provide to a application. For measuring the accurate available bandwidth, they consider the dynamic path selection and different classes service. Even though they design the scheme in real time application, they are suffering from the heavy probe packet stream. PTR [10] focus on searching the sending rate at source node that equals to the arriving rate at destination node. For obtaining the sending rate, PTR finds that the initial packet pair gap affects the arriving rate. Based on the arriving rate, PTR obtains the available bandwidth of the probing link. TOPP [69] is one of the famous probe gap schemes. It is a new end-to-end probing and analysis method through three sub schemes. Using the linear regression scheme, they remove the parameter to estimate the available bandwidth. However, it is suffering from the long probing time because of probing the link and analysing the result.

Probe rate schemes increases the transmitted packets to the destination and takes time to get the available bandwidth of the probing link. Real-time system such as multimedia streaming service is sensitive to the retrieval time of streaming packet because the delayed multimedia streaming data cannot be used in the multimedia presentation. The long probing time in the available bandwidth drops the performance on multimedia streaming service. Probe gap schemes use the interval of consecutive probe packets, since the interval or gap between probe packets is increased in heavy cross traffic: Spruce, IGI and Delphi

Spruce [11] is estimation tool for the measurement of the available bandwidth. For measuring the available bandwidth, it checks the gap between the probe packets. Then, spruce calculates the number of the bytes between two probe packets using inter-probe space. Spruce scheme uses the a poisson process to handle the gap between probe packet. IGI [10] is similar scheme to the PTR that is one of probe rate schemes. IGI also decides the initial gap between probe packets, then gets the available bandwidth of the probing link. However, IGI focus on calculating the background traffic load, while PTR calculates packet receiving rates. During the bandwidth estimation, Delphi [68] use the heuristic and parametric model and queuing theory. It also changed the traffic rate of probe stream depending on the network status. The Delphi estimates the accurate available bandwidth at high cross traffic, while it overestimates the available bandwidth at low cross traffic.

2. Bandwidth Estimation in Wireless Network

The estimation schemes in wired network assume the 'first-come-first-served' for evaluating the available bandwidth, while wireless network environments control the distributed manner. This leads the false expectation to probe the available bandwidth to the link. In multimedia streaming service, underestimation of the link can lose the chance to serve the high quality of multimedia, while overestimation can increase the packet loss and indirect loss that the even transmitted packet cannot be used.

Wireless networks are error-prone and distribute the control scheme to avoid collisions. Therefore, it makes the difficult to detect the exact available bandwidth in a wireless network. In Probegap [28], the idle time in the link is the milestone for bandwidth estimation of a wireless network. They make the graph on the delayed time in received probe packet, then find transition point in the graph to know the available bandwidth. The multimedia streaming service is critical to the delay time. Also, it is difficult to search for the transition point in the graph in heavy cross traffic. TOPP [70, 71] is one scheme of the probe rate in wired networks. They are looking for the input rate that is the same as the output rate to estimate the available bandwidth in wireless network environments. However, they are suffering from long probing times, because of the increased linear traffic of probing packets. SLoPS [72, 27] increases the probing traffic for the fast estimation of available bandwidth. SLoPS is able to estimate the range of the available bandwidth, but it is not accurate. In multimedia streaming services, this inaccuracy estimation decreases the serviced multimedia quality.

3. Cross-layer Scheme

Recent research [73, 74, 75, 76] on mobile multimedia streaming has proven the good performance of a cross layer scheme better than a single layer scheme.

In [77], they provide the embedded real-time schedule in the operating system. In the real-time system, each job has their own dead line. They develop the scheme to handle jobs using the statical distribution. Also they suggest the GRACE-OS to handle the jobs efficiently. [76] presents the voltage scaling algorithm for adjusting multimedia application to CPU speeds in mobile multimedia systems. In the multimedia stream, the frame size is different depending on the frame type and the relationship between frames. When decoder decodes the frame with the low rates, they save the power to use the low voltage. However, they use the high voltage to decode the large frame. [74] provides the adaptation scheme of video quality and the efficient power management scheme through minimizing the energy consumption. They suggest the the low level architectural schemes and OS power-saving schemes. They design the middle ware to provide the power reduction. [73] tunes the system parameter in the unified framework through sublayer interactions using cross-layer schemes. They design the adaptive and unified framework through the feedback information.

In addition, cross layer scheme is used for supporting the optimal power management scheme in wireless multimedia streaming service through cross-layer schemes. [78] analysed the error management scheme through cross-layer scheme considering transmission power and delay. In [79], cross-layer scheme between application-layer and MAC-layer provides the adaptive multimedia transmission scheme in wireless environments. [80] presented the energy efficient error control scheme in multimedia multicast in wireless environment through cross-layer scheme. [81] coordinates three layers and error detection and recovery scheme that presents error detection in the hardware layer, error recovery scheme in middleware and error-resilient encoding in the application layer.

4. Active Intermediate Node

There have been a handful of studies that show how an active intermediate node can improve service quality for multimedia streaming service [82, 83, 84, 85, 86].

Some client can not support the high quality stream because of some limitations such as network or power. In the [82, 83], they suggest that the intermediate node change the quality of the streams using the multimedia transcode. It can save the available bandwidth between server and intermediate node and decrease the response time to the client. In [84, 87], intermediate node increase the efficiency on the streaming service. The late packets cannot be decoded at client and just waste the bandwidth of the link. For the efficient stream service, the late packets should be removed as soon as possible. [84, 87] checks the jitter of the packet based on the packet delay. If some packets exceed the threshold value, an intermediate node eliminates the delay the packet.

For recovering the missed packet, [85] suggests the adaptive retransmission in the streaming service. It is difficult to decide whether packets are retransmitted or not. If retransmitted packets are arrived too lately, they just consume the available bandwidth of the link. To solve the problem, they check the presentation time of the multimedia data in the packet. The proxy retransmits the packets that can be arrived on time. [86] suggests that intermediate node helps the stream service. When the client joins the ongoing broadcasting, it waits the receiving the initial streams. If the intermediate node has the initial streams, they forward the initial stream to the client, clients can play stream without startup delay.

Currently, there are not enough study for handling the environmental variation, such as the available bandwidth or lost packet, have not been studied so far. An STB is also an intermediate node for streaming service and has four major components: a network interface, an MPEG decoder, graphics overlay and an presentation engine [88].

B. Motivation

Wireless Local Area Networks (WLANs) are becoming more and more popular attracting the interest of researchers, system integrators and computer manufacturers. For providing good quality of the multimedia streaming service, variable scheme are used in wireless network including scalable multimedia, bandwidth estimation and collision avoidance.

1. Scalable Streaming Service

In the heterogeneous network environments, scalable multimedia streaming service can provide the acceptable quality to multimedia users through scalable video coding. A traditional scalable streaming service contains spatial scalability and temporal scalability based on the quality base layer and enhancement layer, respectively. The spatial scalable scheme divides the large frame into a small base frame and additional data that improve the quality of the base frame. The temporal scalable scheme groups frames with high frame rate per second (FPS) into several groups of frames with low FPS.

In the recent scalable streaming service, the packet-based scalability allows the fast bit adaptation. The multimedia streaming data are contained into the network abstraction layer (NAL) packets. NAL has three parameters including dependency_id, temporal_id and quality_id that stands for spatial scalability, temporal scalability and SNR scalability, respecively [89].

Multi-layer streams through scalable multimedia streaming service can make the multimedia server to provide the various multimedia clients with different environments and capabilities. Even though scalable multimedia streaming service can provide the several layers streams instead of a large stream, detection schemes are required for the adaptation of scalable service. It is critical for scalable stream services to decide when he quality of multimedia stream is changed.

2. Bandwidth Estimation

Bandwidth estimation is a prerequisite problem for real-time applications in wireless networks. There are two factors making this problem unique. First of all, unlike wired networks, traditional 'first-come-first-served' is not used to schedule bandwidth among connections in wireless networks. To avoid collisions in wireless networks, nodes are arranged in a distributed manner. This arrangement causes bandwidth estimation methods in wired networks using intervals [11, 10, 68] or rates [70, 69, 27, 72] inapplicable for bandwidth estimation in wireless networks. Secondly, they create the probe stream for estimation that consumes the bandwidth for streaming service.

[28, 90] suggested that idle time of a link in a wireless network can be a major milestone for estimating the available bandwidth as follows. In equation (4.1), let C be the capacity of the wireless network. Idle_rate indicates the rate at which the link is idle. Then the AB, available bandwidth, can be obtained by the following product.

$$AB = C \times Idle\_rate \tag{4.1}$$

However, previous schemes [28, 90] using this equation cause too much overhead to be used in a real-time system for the estimation of the available bandwidth. [28] requires the probe stream for bandwidth estimation, and results show multiple incorrect estimated values in heavy traffic. [90] also captures the whole packet in the wireless network and analyzed the captured packet. It requires the heavy CPU load and network I/O operation. For real-time applications such as multimedia streams, it is difficult to use these schemes; therefore, we introduce an efficient scheme to calculate the idle_rate.

3. 802.11 Wireless Standard

The IEEE 802.11 protocol [91, 92] is the dominant standard for WLANs and employs the Distributed Coordination Function (DCF) as the essential Medium Access Control (MAC) method.

a. Network Allocation Vector

DCF defines two access mechanisms to employ packet transmission; the default, two-way handshaking technique called basic access and the optional four-way handshaking RTS/CTS reservation scheme. The RTS/CTS scheme involves the transmission of the short request-to-send (RTS) and clear-to-send (CTS). It controls packets prior to the transmission of the actual data packet. Since collisions may occur only on the RTS packets and are detected by the lack of the CTS response, the RTS/CTS scheme results in an increase on system performance by reducing the duration of collisions, especially when long data packets are transmitted. The RTS/CTS scheme is also employed to result in a better performance in the presence of hidden stations. When two nodes in a wireless network share the same access point (AP) but cannot hear each other, one node will not be able to know whether the other node is already using shared resource, that is, the wireless channel. For addressing this hidden node problem, each node uses the NAV that shows how long other nodes allocate the link in the IEEE 802. DCF MAC protocol. Even though a node is located at a place where it cannot reach other active nodes, the node can know whether another node is already using the wireless network by checking its NAV. In Fig. 22, when the sender sends RTS request to send (RTS) to the receiver (AP), *Other-1* node that is reachable from sender updates its NAV. However, *Other-2* node does not update NAV, because it is not reachable from sender. When the receiver sends clear to send (CTS), *Other-2* node updates its NAV. The idle time in the wireless network can then be estimated from the NAV information.

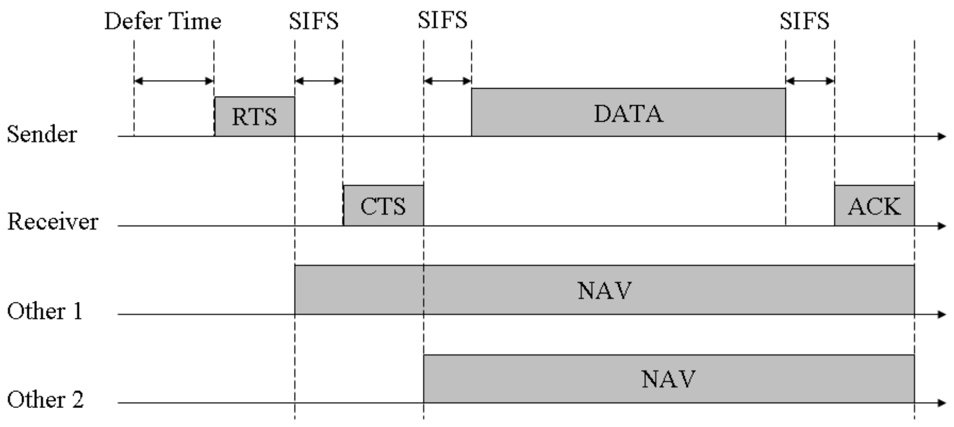

Fig. 22. Network Allocation Vector

b. Multiple Transmission Rate

Based on channel coding schemes, the physical layers (PHY) supports multiple transmission rates. In 802.11b [92], PHY provides four transmission rates of 1Mbps, 2Mbps, 5.5Mbps and 11Mbps, while eight transmission rates including 6Mbps, 9Mbps, 12Mbps, 18Mbps, 24Mbps, 36Mbps, 48Mbps and 54Mbps are supported by PHY in 802.11a [91]. Even though 802.11 determine the various transmission rates in each standard, negotiation of transmission rate is not specified. Two schemes are widely adapted in transmission rate adjustment. One scheme provides transmission adaptation based on Signal to Noise Ration (SNR) [93, 94]. Other one is based on the history of the acknowledgment (ACK) reception that is suggested by Lucent Technologies WaveLan-II [95]. In the failure of receiving two consecutive ACK frames, the negotiated transmission rate is lower.

C. *IdleGap*

All nodes in a WLAN share the same resource; that is, a wireless channel. If a node in a WLAN is utilizing the resource, the additional node(s) should await the release of the wireless channel. The whole bandwidth is consumed by other nodes or available to any node in a wireless network in equation (4.2).

$$\begin{aligned} B_{total} &= B_{available} + B_{consumed} \\ B_{consumed} &= \sum_{i=0}^{n} B(i) \end{aligned} \tag{4.2}$$

where B_{total}, $B_{available}$, $B_{consumed}$, n and $B(i)$ is the total bandwidth, available bandwidth, consumed bandwidth, number of node in wireless networks and the consuming bandwidth by *nodei*, respectively. For multimedia streaming service, it is critical to know how much bandwidth is allowed. When $Nodei$ received the multimedia streaming service, $Nodei$ can consume the $B(i) + B_{available}$ bandwidth. $B(i)$ can be known as how many data can be received from multimedia server.

However, it is difficult to estimate the available bandwidth in a wireless networks. In equation (4.1), the capacity and idle rate of wireless link is critical to estimate the available bandwidth. Unlike the wired network, wireless networks like IEEE 802.11 supports multiple transmission rate depending on the environments including error rate and location. Also, the hidden node problem makes it difficult the estimation of the idle rate in wireless link.

1. Network Idle Vector (NIV)

To estimate the idle rate in wireless networks, we build the vector in the MAC layer. When node transmits packet through IEEE 802 protocols stack, each packet includes time information that shows how much time the link is allocated. NAV table stored the information when the link will be available. However, it is not considered about the duration time of link allocation. In addition, NAV is updated only when other node transmit the packet. For the estimation of idle rate, we embedded the network idle vector (NIV) in MAC layer for storing busy time of the link. When packets are received from other nodes or are transmitted, NIV update the duration time in a time slot. N slots in NIV contain the busy time of the link at each time slot. In Fig. 23, AP can reach the whole nodes, while Node 1 cannot receive the packet from Node 3, directly. After receiving the RTS packet at time $r0$, Node 1 knows that AP allocates link for packet transmission from $r0$ to $r1$. During the time from $t0$ to $t1$, the idle time of the link is $r0 - t0$. After finishing the data transmission from AP to Node1, Node 3 transmits the data into AP. Node 1 is located outside of Node 3's transmission range, therefore RTS from Node 3 can not reach into Node 1. Node 1 does not update the NIV table into busy status during time $r2$. After receiving the CTS from AP, Node 1 changes the status of NIV table at duration time $(r4 - r3)$. In the time from $t2$ to $t1$, idle rate is $(r3 - r1)/(t2 - t1)$. The link in the wireless network is used during the whole time $t2$ to $t3$, Node 2 and AP use the link from $r4$ to $t3$ during the time from $r4$ to $r3$.

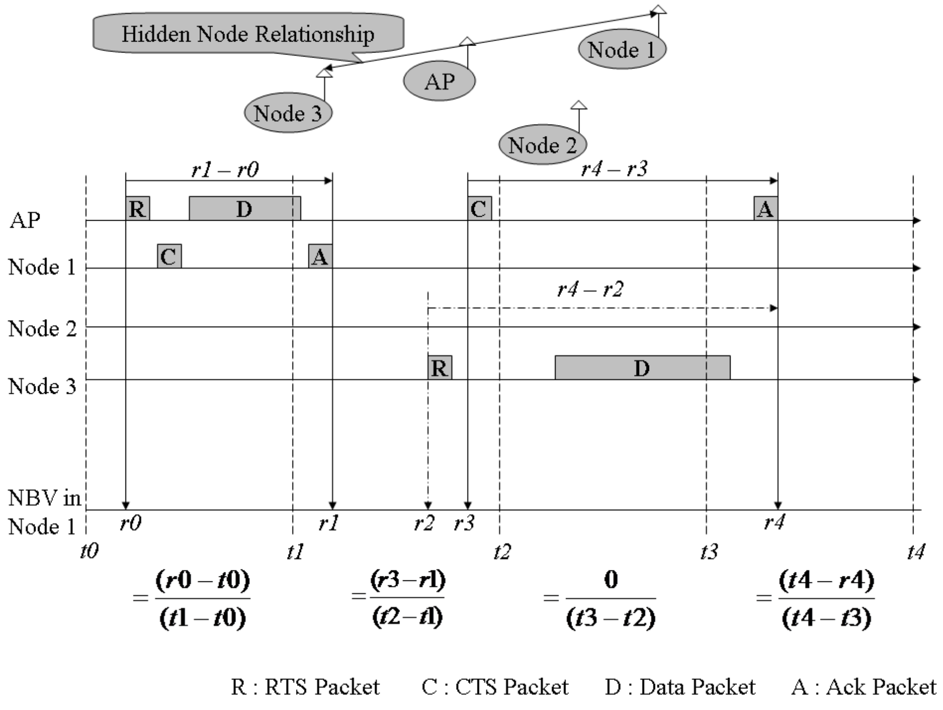

Fig. 23. Network Idle Vector

2. Estimation Scheme

a. Capacity

IEEE 802.11 like IEEE 802.11a, IEEE 802.11b or IEEE 802.11g provides multiple transmission rate. In addition, collision avoidance scheme requires management time for packet transactions including RTS, CTS, Defer Time and short inter frame space (SIFS). In previous works [11, 10, 68, 70, 69, 27, 72, 28, 90], they consider the capacity of link as physical transmission rate. However, 802.11 protocols use the management packets for avoiding the collision. It takes time to transmit and receive management packets. Therefore, the capacity can be changed the packet size and error rate. It requires the adaptive scheme for the measurement of capacity in wireless link.

The duration time, RT_i, in the 802.11 protocols are calculated in the Mac layer for

RTS/CTS scheme. Our module, Idle_Module, retrieves the data size, RS_i, whenever packets are coming. Then, our Idle_Module measures the dynamic capacity in wireless link using equation (4.3).

$$C = \frac{RS_i}{RT_i} \tag{4.3}$$

b. Idle Rate

All nodes in a WLAN share the same resource; that is, a wireless channel. If a node in a WLAN use wireless channel, the additional node(s) should await the release of the wireless channel. During a transmission in a WLAN, a node can be one of the following: sender, receiver, or onlooker. If a node transmits data to another node, it is a sender. A node is a receiver if receives data. Finally, when a node does not join the transmission, it is an onlooker.

The busy time of the link can be estimated by adding up all the transactions of nodes in the network as depicted in equation (4.4). Here T_l is the busy time of link l and *TT (i, j)* indicates the transaction time between nodes i and *j*.

$$T_l = \frac{1}{2} \times \sum_{i=1}^{n} \sum_{j=1}^{n} (TT(i,j)) \tag{4.4}$$

Unfortunately, we cannot know all the transaction times form the nodes in the network. In addition, obtaining the transaction information can increase network traffic, hence affecting current traffic on the network. Therefore, we propose to obtain all the necessary information from one node in the network as follows. The transaction time of node i can be obtained via the sum of the sending and receiving times to/from node i ($TT(i,j) = ST_i + RT_i$, where ST_i is the sending time from node i to j and RT_i is the receiving time from node j to i). though a node is located at a place where it cannot reach For the transaction time between other nodes, we can get theonlooking time from the NAV in node i that is updated in other node transactions ($TT(i,j) = OT_i$, where OT_i is the onlooking time at node i). Therefore, we can estimate the busy time T_l in any node i in the network as shown in 4.5.

$$T_l = ST_i + RT_i + OT_i \tag{4.5}$$

$$Idle\_rate = 1 - \frac{busytime}{totalelapsedtime} \tag{4.6}$$

We can then obtain Idle_rate using the busy time:

3. System Model

We propose to add an Idle_Module in the MAC layer of a node in the network. This module obtains the busy time T_l from (a) and (b) in Fig. 24. The transaction time of node can be obtained through accessing outgoing and incoming packets $ST_i + RT_i$ between the network layer and the link/mac layer in Fig. 24. The update process of the NAV triggers the Idle_Module to update its value. An application can access the Idle_Module to get the idle rate, $1 - (busytime)/(totalelapsedtime)$. Then applying the idle rate and link capacity C in equation (4.3) , the estimated bandwidth of the link can be calculated with minimal effort.

D. *ActiveSTB*

ActiveSTB acts as gateway to downstream heterogeneous clients and also performs quality-adaptation to various network bandwidths, while simultaneously forwarding the buffered stream allowing a client to view quality playback of multimedia stream. *ActiveSTB* is designed to efficiently manage the wireless transmission of the multimedia stream by early dropping of useless data and estimating the available bandwidth.

The removal of useless multimedia streaming data from the *ActiveSTB* cache

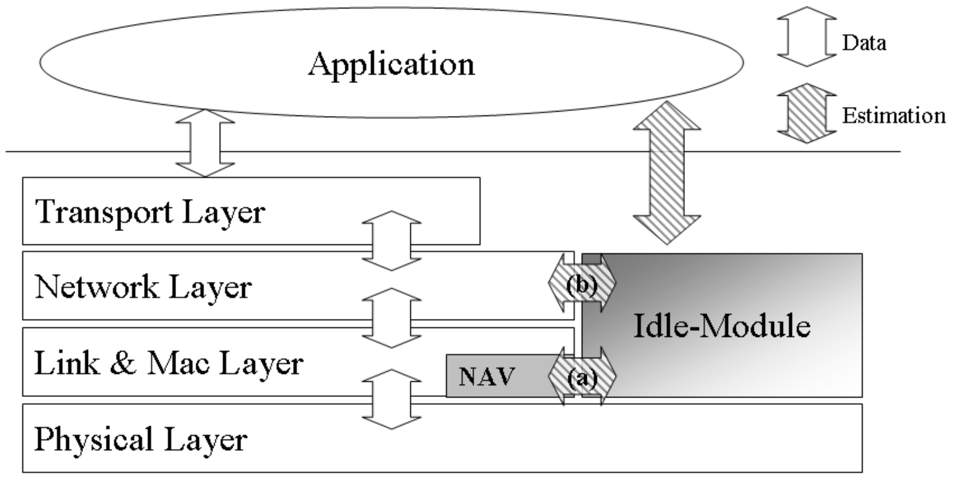

Fig. 24. *IdleGap*

can save the bandwidth of network. Loss in scalable streaming service has two classifications: indirect loss and direct loss. For efficiently managing the wireless link, our *ActiveSTB* saves the bandwidth of wireless link by eliminating indirect loss that packet are transmitted not decoded.

1. Drop of Residual Data

In the scalable multimedia streaming, each layer has the different priority. Base layer stream has high priority, while enhancement layer stream has low priority in scalable stream service. When the available bandwidth cannot accommodate whole frames in scalable stream, some enhancement layers are removed for smoothing the scalable multimedia service. An high quality stream service can cause the congestion in wireless channel and make the unexpected delay in multimedia delivery, while an low-quality streaming service lose the chance to provide the better service to the user. Our active scheme determines the adaptive quality for scalable streams in wireless channel.

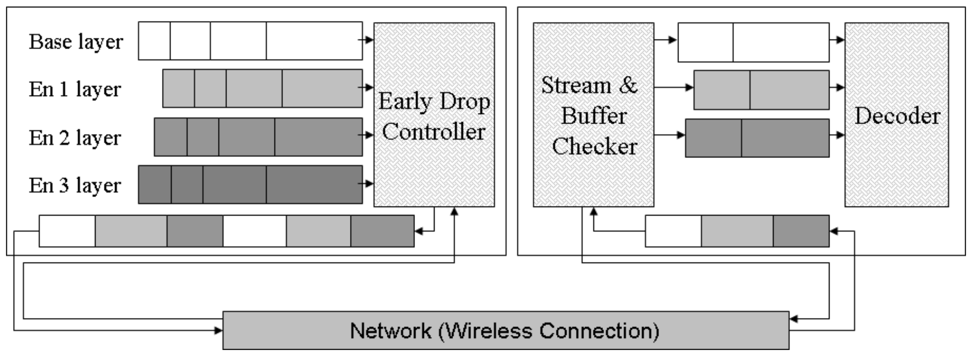

Fig. 25. Early Drop for Residual Multimedia Stream

Fig. 25 shows the architecture for the adaptive scalable multimedia streams. In Fig. 25, the stream controller provides the feedback information on the status of estimated bandwidth, EB_i, and cached stream. Based on the information, the early drop controller decides how much data are transferred into client. Each GoV in the scalable streams has their own time stamp for decoding, TS. The time stamp at the initial GoV in the streams are configured as 0. Our *ActiveSTB* extracts the time information from the packets. It takes time to deliver the packets into client, DN_i. The client starts to decode the received packets after buffering the initial GoVs, B_{init}. We assume that gap between GoV is same, T. For playing scalable stream at PT_i, the client should receive them before access them, RT_i.

$$RT_i < PT_i \tag{4.7}$$

For satisfying fomula 4.7, scalable streams are delivered into client before the cached stream in the buffer are running out. Our *ActiveSTB* transmits each stream at ST_i into client considering the decoding time, TS. Also, decoder consumes the GoV considering the decoding time, j. So, equation (4.7) is driven into equation

(4.8).

$$\begin{aligned} DN_i &< B_{init} - RT_0 && \{StartupDelay\} \\ DN_i &< PT_{i-1} - PT_j && \{PlayingVideo\} \\ &< PT_0 + TS \times \{i-1\} - PT_0 - TS \times j \\ &< TS \times \{i-j-1\} \end{aligned} \tag{4.8}$$

In equation (4.8), DN_i can be calculated by multiplying size of multimedia data, SM_i, and estimated available bandwidth, EB_i.

$$\begin{aligned} SM_i &< \frac{B_{init} - RT_0}{EB_i} && \{StartupDelay\} \\ SM_i &< \frac{TS \times \{i-j-1\}}{EB_i} && \{PlayingVideo\} \end{aligned} \tag{4.9}$$

Therefore, multimedia stream that satisfies equation (4.9) can be arrived on the client side before decoding streams.

2. Drop of Corrupted Data

In Fig. 25, each GoV in the scalable multimedia stream contains four layers that include a base layer and three enhancement layers. Decoding a layer requires the referred other layers because of the hierarchical relationship between layers. When the *ActiveSTB* notices that third layer in a GoV is incomplete, so both the third and the forth layers in the GoV are early dropped in the STB and not forwarded to client. This early dropping decreases bandwidth consumption by eliminating corrupted multimedia data.

QoS of streaming multimedia data is determined by not only the amount of direct loss but also one of indirect loss. The *ActiveSTB* transmits the only acceptable multimedia data. Table VI shows the variables used for stream packet transmission and validation equations.

Table VI. Packet Transmission and Validation Calculation Variables

G_g	The g_{th} GoV in the multimedia stream
$L_{g,l}$	The l_{th} layer at the g_{th} GoV
$P_{g,l,s}$	The s_{th} packet in the l_{th} layer at the g_{th} GoV
NG	Number of GoVs in multimedia stream
NL_g	Number of layers in GoV(g)
NP_{gl}	Number of packets in the l_{th} layer at the g_{th} GoV
PLR	Packet loss rate
$Size_{gl}$	Size of the l_{th} layer at the g_{th} GoV

In equation (4.10), the server divides l layer in g GoV multimedia stream into several packets for transmitting:

$$L_{gl} = \bigcup_{i=0}^{NP_{gl}} P_{gli} \tag{4.10}$$

When a layer is complete with all available packets, a client can decode the received layers. One packet loss can cause other packets in the same layer and referring layer to be thrown away. The safe transmission of all packets in l layer ensures that the multimedia data at l layer is valid for decoding as in equation (4.11).

$$Valid\_network(L_{gl}) = \prod_{i=0}^{NP_{gl}} (1 - PLR(P_{gli})) \tag{4.11}$$

Each layer has a hierarchical relationship with other layers as shown as in equation

(4.12):

$$Valid\_decode(L_{gl}) = \prod_{j=0}^{L_{gl}} \prod_{i=0}^{NP_{gl}} (1 - PLR(P_{gli})) \tag{4.12}$$

To reduce a waste of available bandwidth on wireless channel, we filter out corrupted layers before transmission over the shared wireless channel as shown in equation (4.14), while complete layers are transmitted to a client through wireless channel as shown in equation (4.13):

$$\begin{aligned} Complete(NG) &= \sum_{g=0}^{NG}(\sum_{l=0}^{NL_g}(Size(L) \\ &\times \prod_{j=0}^{L_{gl}} \prod_{i=0}^{NP_{gl}} (1 - PLR(P_{gli})))) \end{aligned} \tag{4.13}$$

$$\begin{aligned} Filtered(NG) &= \sum_{g=0}^{NG}(\sum_{l=0}^{NL_g}(Size(L) \\ &\times (1 - \prod_{j=0}^{L_{gl}} \prod_{i=0}^{NP_{gl}} (1 - PLR(P_{gli}))))) \end{aligned} \tag{4.14}$$

E. Experimental Results

To evaluate the performance of our *IdleGap* method, tests were run using the NS-2 simulator. As shown in Fig. 26, there are seven nodes including three wired nodes, three wireless nodes and an AP. In the wired network, the capacity of the link was set to 100Mbps, while the capacity in wireless network was set to 1 or 10Mbps.

1. Scalable Multimedia

Table VII shows five H.264 streams for the simulation. Five streams include three movie trailers and two scenic video clips. The streams are divided into five or six sub streams using Joint Scalable Video Model (JSVM) codes [101]. During the creation of sub layers from one streams, we configured three factors including quantization,

Table VII. Scalable Streams

Frame Name	Encoding Type	Number of Frames
Amazing Caves [96]	Scenario 1	2031
Bourne Ultimatum [97]	Scenario 1	2125
I Am Legend [98]	Scenario 1	2397
Simpsons [99]	Scenario 2	1568
To The Limit [100]	Scenario 2	919

space and frame rate.

Table VIII. Scalable Stream Scenario 1

	QP	Frame Rates	Frame Size
Layer 0	38.0	15	320×240
Layer 1	32.0	30	320×240
Layer 2	30.0	30(15)	320×240 (640×480)
Layer 3	28.0	30	640×480
Layer 4	26.0	30(15)	640×480 (1280×960)

We create five scalable stream H.264 video following two scenarios. Table VIII shows the scenario 1. We divided one stream into five sub streams. Base stream contains 320×240 frames, 15 frame per second and 38 quantization value. When the whole enhanced layers are added into scalable streams, quality of video is enhanced upto 1280×960 frames, 30 frames per second and 26 quantization value. The size of frame in layer 2 is larger than one in layer 1, while the frame rate is lower thanlayer 1. When layer 0, 1 and 2 layers are decoded, some frames are decoded at larger size (640 × 480) and other frames are decoded at small size (320 × 240). The layer 4 in scenario 1 gets lower frame rate than layer 3 in scenario 1. In scenario 1, we create high quality scalable streams, low quantization value means that non-zero Huffman codes is increasing. In addition, large frame size also multiplied the size of scalable stream. However, it can provide the high quality of scalable stream service.

Table IX. Scalable Stream Scenario 2

	QP	Frame Rates	Frame Size
Layer 0	38.0	15	160 × 120
Layer 1	38.0	30	160 × 120
Layer 2	38.0	30(15)	160 × 120 (320 × 240)
Layer 3	34.0	30	320 × 240
Layer 4	34.0	30(15)	320 × 240 (640 × 480)
Layer 5	32.0	30	640 × 480

Table IX shows the scenario 2 that makes low quality streams with high quantization values and small frame size. It can increase the probability to send the frame when the network are congested. '*Amazing Caves*', '*Bourne Ultimatum*' and '*I am legned*' are encoded with high quality followed by scenario 1, while '*Simpsons*' and '*To The Limit*' are encoded into low quality streams.

2. Simulation Configuration

In Fig. 26, the AP involves three connections: the Wired Node 1 to the Wireless Node 2, the Wired Node 2 to the Wireless Node 1, and the Wired Node 3 to the Wireless Node 3. The Wired Nodes 1 and 2 generate the cross traffic. The Wired Node 3 transmits stream dta into the Wireless Node 3. We evaluate the estimation scheme including *IdleGap*, ProbeGap, Pathload and Spruce. The three step simulation for

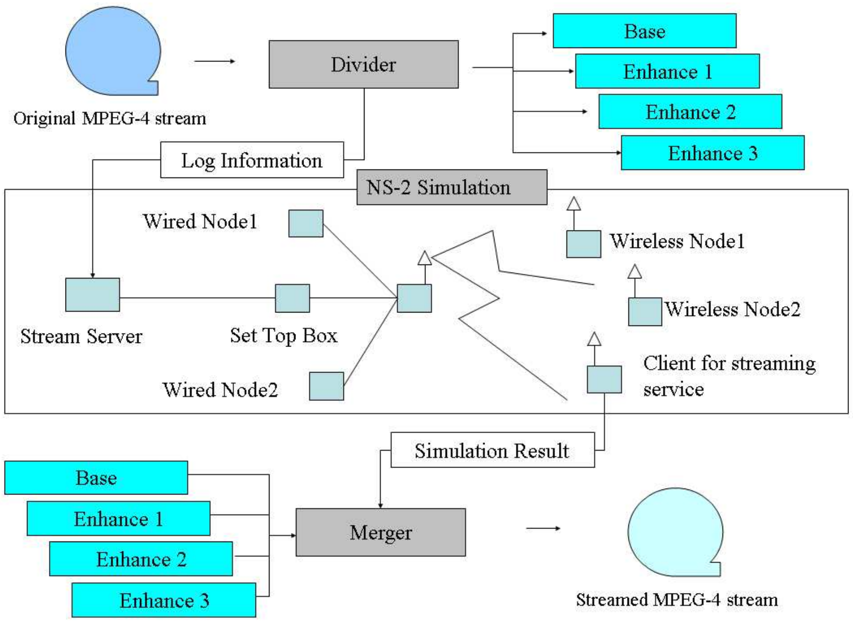

Fig. 26. Simulation Environment

streaming service is conducted using JSVM, merger and NS-2 simulator. First of all, JSVM divides the original stream into several layers, and then logs the size and decoding time information of each layer in terms of GoV. Secondly, the NS-2 network simulation is conducted using the log file generated by JSVM, and tracks when the packets are arrived into the client. Lastly, our merger decides whether packets are valid using the arrival time of packets, the decoding time of frames and startup delay. At the network simulation step, we have three connections including streaming service and two cross traffic. Multimedia server transmits the multimedia data into STB that caches and forwards the cached data into clients using wireless connection. The maximum of bandwidth of wireless channel is 10 Mbps.

3. Streaming Service

For reducing the indirect loss, it is critical to detect which packets contain the corrupted frame. MPEG Standards provide the start code and resync code for skipping the corrupted data. However, STB should scan the whole received packets for finding the start code and resync code. It creates too much overhead on the STB. When multimedia server divides the frame into the packets, we add more information including frame_no, frame_seq, layer_id and frame_flag in the header of the packet. frame_no, frame_seq and layer_id show the frame number, sequence number and stream id. The frame_flag stands for the status of the frame. '0', '1' and '2' in the frame_flag means the middle, first and last packet in the frame, respectively.

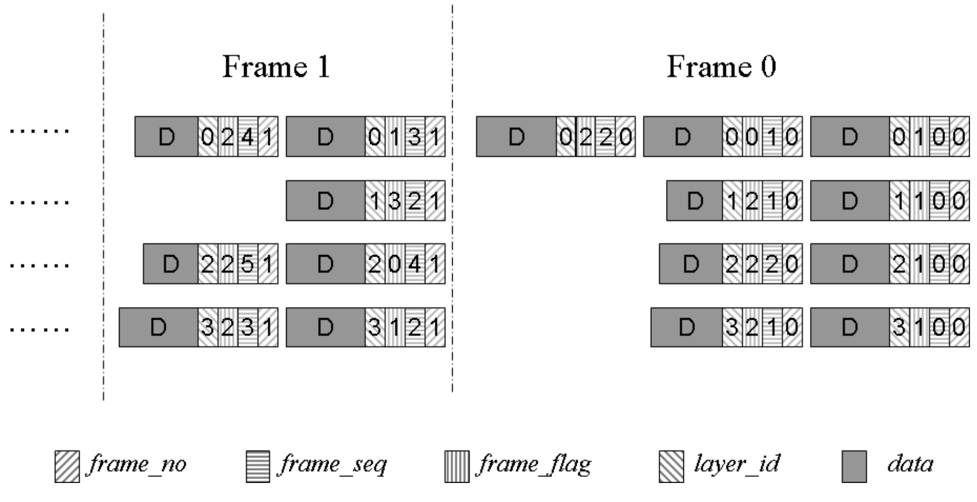

Fig. 27. Active STB Packet Management

In Fig. 27, the first, second, third, forth and fifth field in the header shows frame_no, frame_seq, layer_id, frame_flag and data, respectively. For inspecting the packets, *ActiveSTB* checks whether to set the first bit (01) in the first packet and the second bit (10) in the last packet in the layer. After checking first and second packets, the *ActiveSTB* scans the whole middle packets in the layer using frame_seq field. If the *ActiveSTB* finds layers where some packets are missing, the layer with missing packet and sub layers eliminated from buffer in the *ActiveSTB*. In Fig. 27, the third layer in frame 0 miss the packet with frame_seq 1. Therefore, the first layer and second layer in the frame are transmitted to the client through wireless link. In the frame 1, the first bit in the frame_flag of the first packet at the third layer is not set. Only the first and second layers are transmitted. The first packet in the second layer set the first and second bits (11) at the same time. It means that the packet contains the whole data in the second layer in frame 1.

4. *IdleGap*

We evaluate 4 estimation schemes and none-estimation scheme in multimedia streaming service in wireless network. None stands for none-estimation scheme that the whole cached streams are forwarded into the client. Spruce, Pathload and ProbeGap are stands for spruce scheme, probegap scheme and pathload scheme, respectively.

Fig. 28 shows the estimated available bandwidth value for each algorithm. The capacity of the wireless network in our simulation is 10 Mbps. Probing time for each algorithm is 4000 seconds and 800 probing packets are allowed. In light cross-traffic, the ProbeGap and Sprce produce bandwidth estimation reflective of measured available bandwidth values. However, they provides the unacceptable results in heavy-cross traffic. The ProbeGap shows multiple transition points over 4 Mbps cross-traffic. The estimated results fluctuate wildly in heavy cross traffic. In [11], the intra pair gap is set to the transmission time of the narrow link. This causes the underestimation of the available bandwidth for the link. The intra pair gap is calibrated to

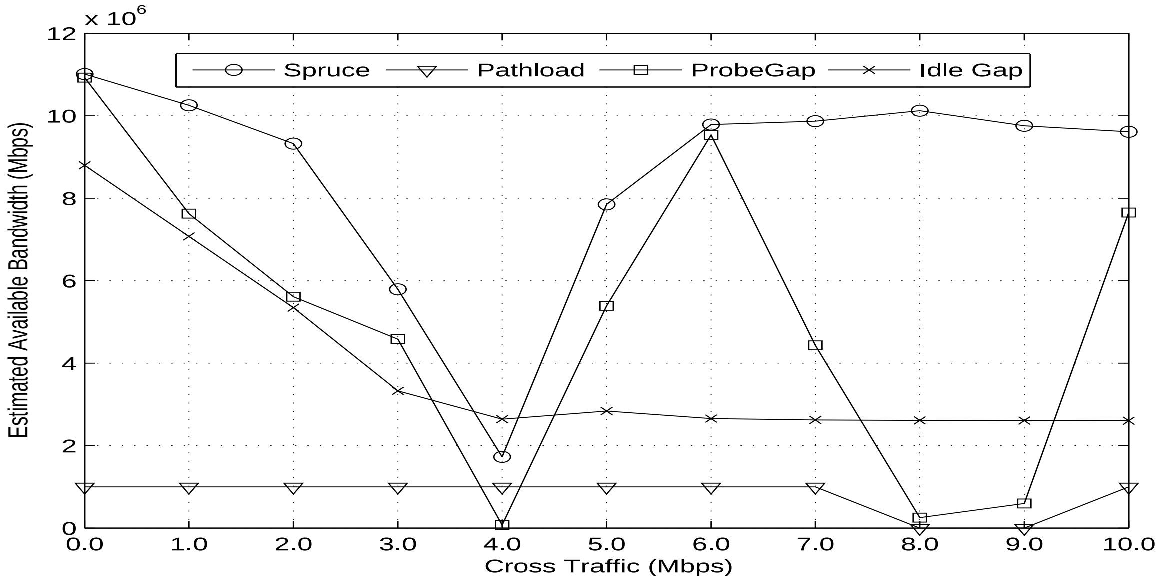

Fig. 28. Estimated Bandwidth with Cross Traffic

reflect the available 4.0 Mbps with no cross-traffic. Even after the calibration, Spruce overestimates the bandwidth severely with more than 4.0 Mbps cross-traffic. Wireless communication requires lots of management packets, therefore available bandwidth are reduced. It makes the high drop rates of probe packets. Thus, the estimated bandwidth value becomes polluted. This cause the overestimation of the available bandwidth. The *IdleGap*, which uses NIV to estimate bandwidth, shows the closest match to the real bandwidth. After 4.0 Mbps cross-traffic, the saturation of wireless link occurs due to the overhead of the wireless network such as defer time and RTS/CTS.

In Fig. 29, the estimated idle times in the AP and node 3 are depicted with different packet sizes of the same cross-traffic. Cross-traffic in the simulation is 1 Mbps that consumes 10 percent of capacity in 802.11b and the packet size is changed from 128 to 1024 bytes. We observe that packet sizes between 512 and 896 bytes provide more accurate estimation. The estimated idle time with the small size packet

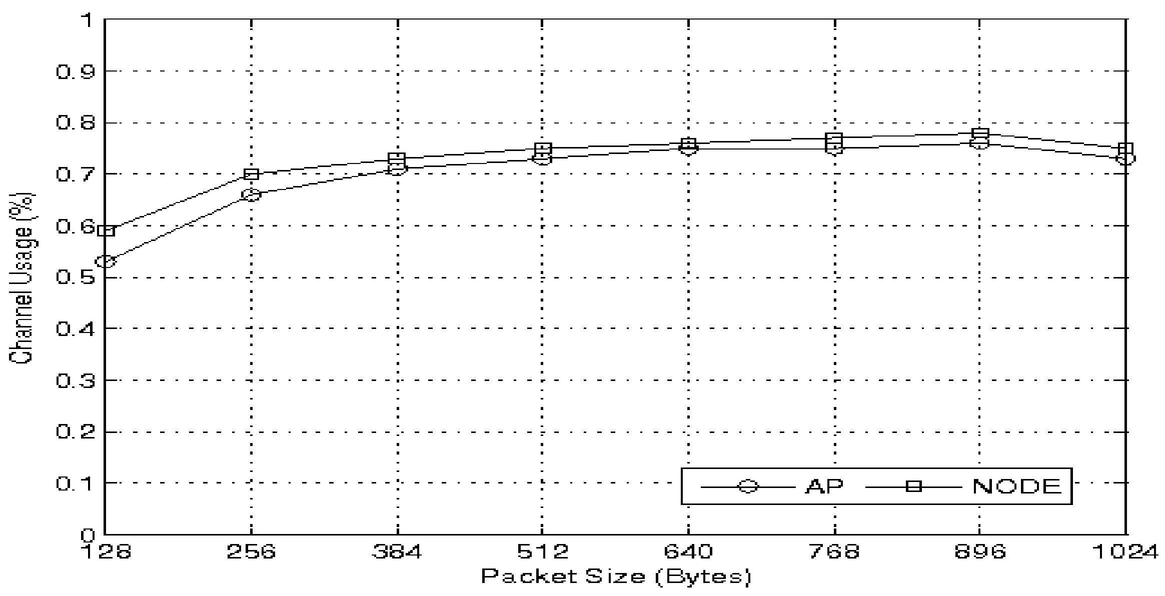

Fig. 29. Estimated Bandwidth with Different Packet Sizes

is smaller than the one with the large size packet. In order to transmit a packet, the sender should send the RTS, CTS, and ACK to the receiver. The frequent transmission of small packets increases this overhead. That is why the *IdleGap* underestimates the available bandwidth with small size packets. On the other hand, with the largest size packets (1024 bytes), the estimated idle time is also decreased slightly. During the transmission, the large packet is broken into several fragments in the Mac layer to reduce the error rate, which again causes overhead. Estimated bandwidth in the AP inclines to be smaller than the one in node 3. If node 3 is a hidden node, it receives only the CTS, not the RTS. Then, node 3 cannot detect the busy time gap between the RTS and the CTS.

5. *ActiveSTB*

When cross traffic exceeds into 4 Mbps. Most multimedia data are not transmitted. Even though they are transmitted, the cannot be used at the decoder because of the severe indirect loss. Therefore, we consider only when cross traffic are lower than 4 Mbps.

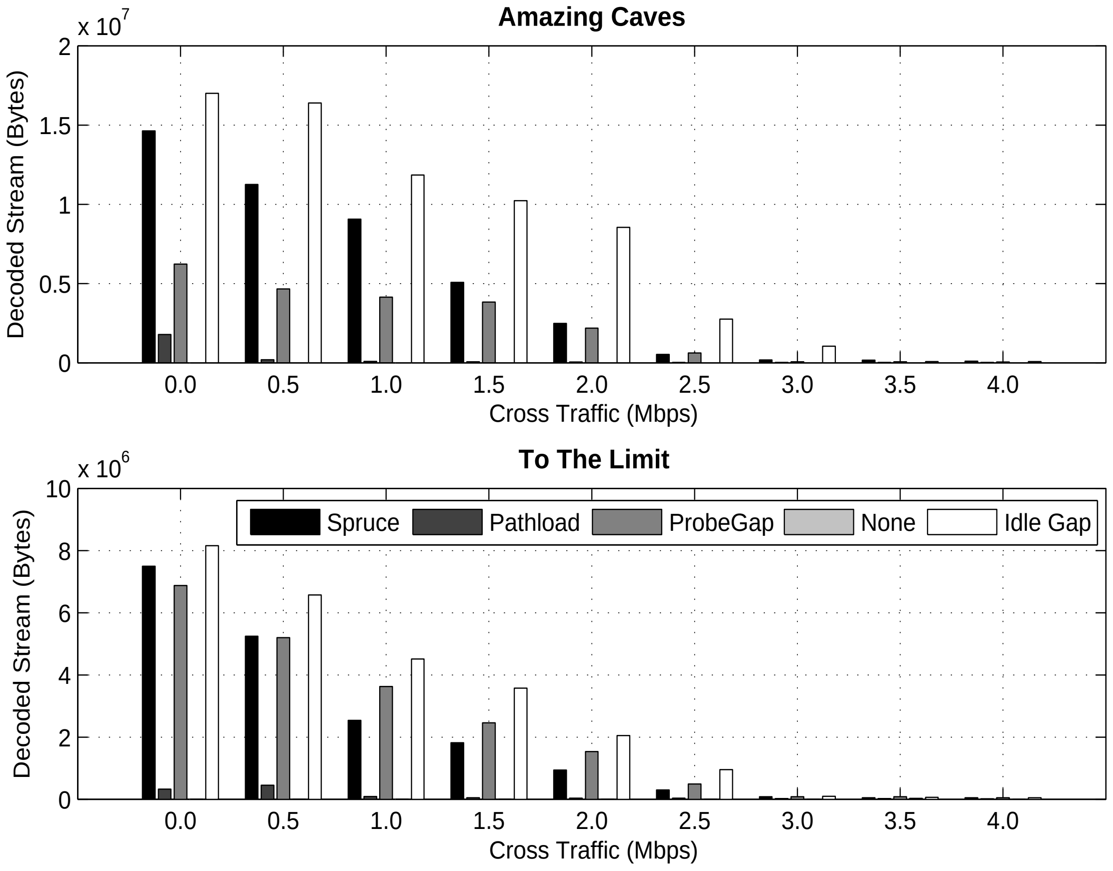

Fig. 30. The Size of Decoded Streams (*Amazing Caves*, *To The Limit*)

Fig. 30 shows how many multimedia data are decoded at the client. X-axis shows cross traffic, while Y-axis stands for the size of decoded stream. The None estimation scheme transmits the whole layer into the client. Wireless channel cannot support the bandwidth to transmit the layers. Therefore, most of them cannot arrive before decoder use them. The Pathload estimation scheme creates the heavy probe packet streams to calculate the available bandwidth. Probe packet stream consumes the bandwidth even for streaming service. The Spruce and the ProbeGap schemes show better results than the pathload scheme. Their probe packet streams are not heavier than the pathload scheme, so they provides high quality multimedia service into a client. In the *Amazing Caves* and the *Bourne Ultimatum* streams, the Spruce scheme shows the better results than the ProbeGap scheme. However, the ProbeGap scheme can decode more data than the Spruce scheme in the *Simpson* and the *To The Limit* streams. The Spruce scheme overestimates the available bandwidth in the wireless link. It can give the chance to transmit the more multimedia data in low workload. However, it can increase network congestion at the heavy workload or heavy cross traffic.

The *ActiveSTB* drops some multimedia streams based on the estimated available bandwidth. After receiving the feedback from the client, the *ActiveSTB* checks the size of the cached multimedia, and then decided how much data are transmitted into the client. Much streaming data creates the congestion in wireless network, but low streaming data lose the chance to improve the quality of multimedia streaming service. The results using other streams except *Amazing Caves* and *To The Limit* show in Appendix B.

In Fig. 31, x-axis shows the cross traffic and y-axis is for how much data are dropped into the client. The 'None' estimation scheme transmits the whole cached stream into the client, so the *ActiveSTB* does not drop the cached stream. In most cases, the Pathload scheme underestimate the available bandwidth in wireless channel. It cause to drop more multimedia stream data. The Spruce scheme overestimates the available bandwidth and drops the small multimedia stream data. However, it cause the network congestion and decrease the size of the decoded stream data. In the *Amazing Caves* stream, the Spruce scheme sends more data to the client and it increases the chance to improve the quality of multimedia streams. The Spruce scheme also sends more data to the client in the *Simpsons* stream. However, it decreases the quality and decoded data. The ProbeGap scheme transmits the small

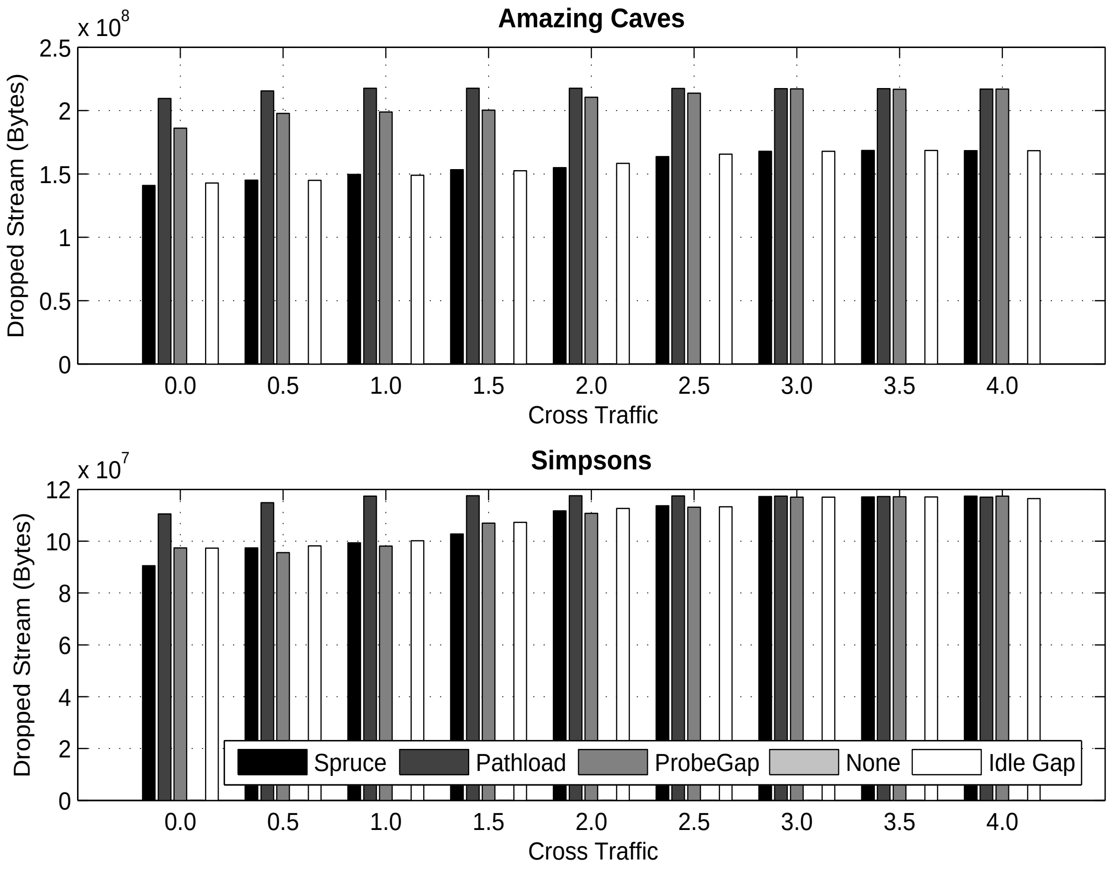

Fig. 31. The Early Dropped Stream (*Amazing Caves*, *Simpsons*)

multimedia data into the client at 0.0 cross traffic because of the underestimation for the available bandwidth. The results using other streams except *Amazing Caves* and *Simpsons* show in Appendix B.

Fig. 32 shows how many multimedia data are lost, even though packet is arriving successfully. When stream data has dependency between other data. For example, stream data including B frames requires the previous and future I or P frames. In other case, packets are not arriving before decoder decode them. This indirect loss waste the available bandwidth and decrease the performance on wireless network. In Fig. 32, x axis shows the cross traffic and y axis is for how many multimedia data are lost indirectly. The ProbeGap scheme underestimate the available bandwidth and

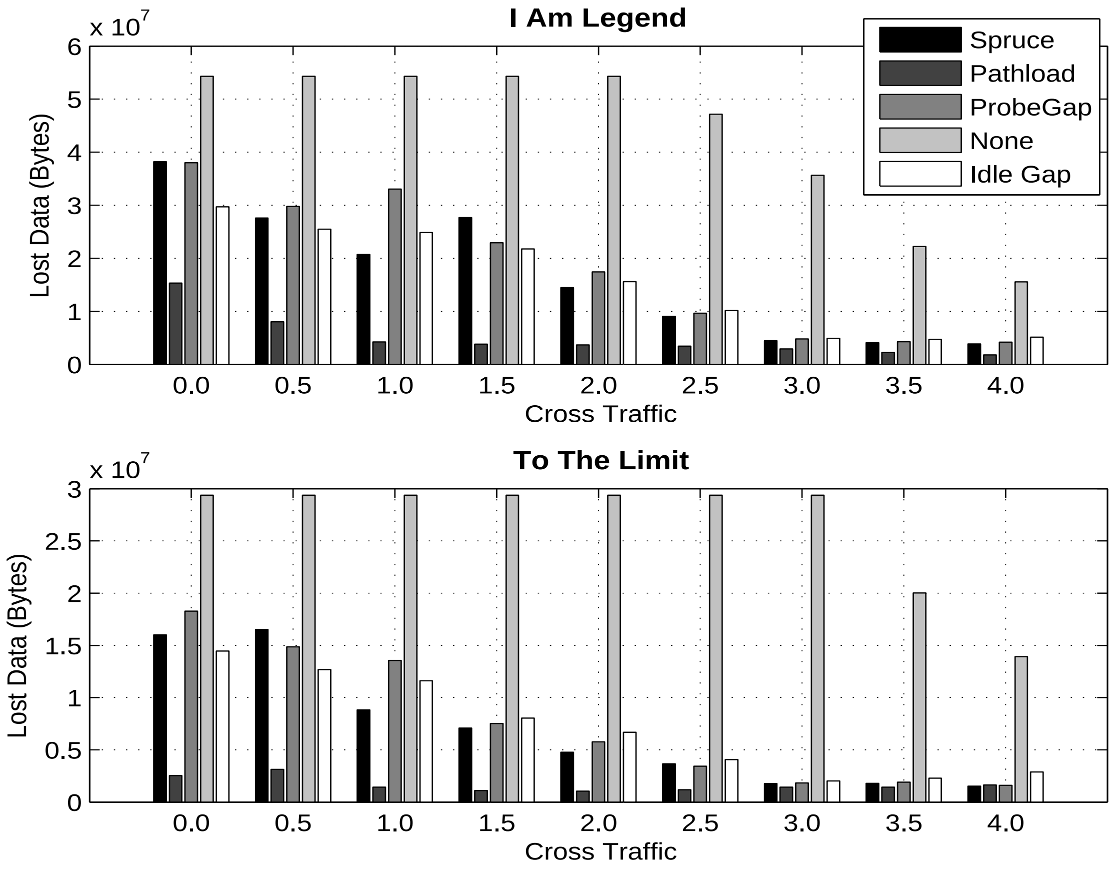

Fig. 32. The Indirect Loss (*I Am Legend*, *To The Limit*)

small data are transmitted into the client at Fig. 31. Therefore, the indirect loss are smaller than other estimation scheme. The None estimation scheme lose most of transmitted data because the data are not arrived on time. Our *IdleGap* scheme shows the low indirect loss. The results using other streams except *I Am Legend* and *To The Limit* show in Appendix B.

Fig. 33 shows the ratio between sent data from an *ActiveSTB* and decoded data at client side. X-axis shows the cross traffic, while y-axis stands for the ratio between the size of sent data and the decoded data. The ratio is critical to manage the performance in wireless network. Even though only small data are decoded, inefficient multimedia steam service can consume lots of available bandwidth. It can decrease

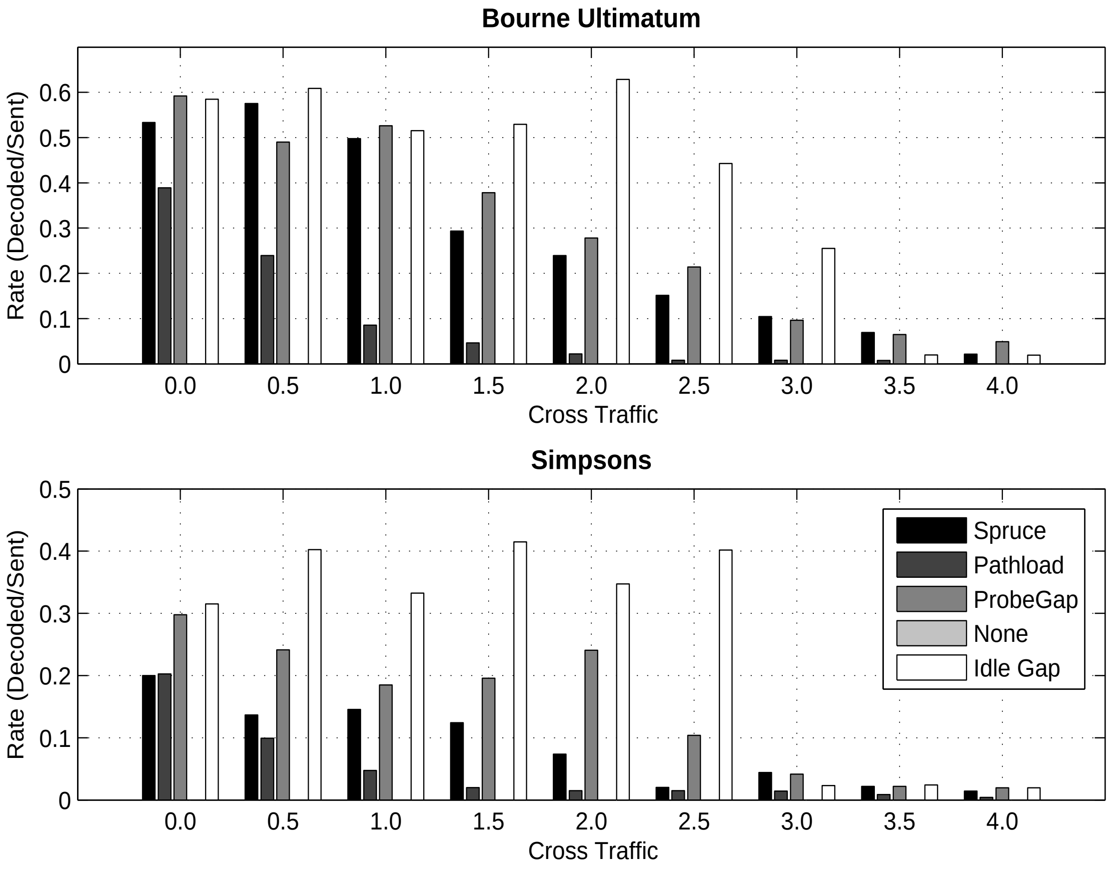

Fig. 33. The Ratio between Sent Size and Decoded Size (*Bourne Ultimatum*, *Simpsons*)

the available bandwidth in other connections.

Our *IdleGap* scheme shows higher ratio than other estimation scheme. The *IdleGap* does not waste the allocated bandwidth. In the None scheme, most of multimedia data are lost because the wireless network with 11 mpbs cannot support the whole original multimedia data such as multimedia streams with large frame size. The Spruce scheme shows the high ratio in the *Amazing Caves* and *Bourne Ultimatum.* In two streams, the Spruce scheme transmits many multimedia stream data before decoding them. However, overestimation of the Spruce scheme in *Simpsons* decreases the ratio. When cross traffic is 2.0 or 2.5 Mbps, the *IdleGap* scheme shows the high efficiency, because the *IdleGap* scheme reduces the suitable size of the transmitted data. The results using other streams except *Bourne Ultimatum* and *Simpsons* show in Appendix B.

In addition, our *IdleGap* scheme outperforms other schemes under overall startup delay. The results of the simulation are presented at Appendix C. The Pathload scheme still the low decoded sizes of stream in the client side because of the heavy probe stream. Long startup delay increases the size of the decoded stream in the client side.

CHAPTER V

CONCLUSIONS

Rapid growth in the number of web users and current HTTP frameworks incurs the overhead in web-based system. Web cluster systems provides excellent and cost-effective solution for this problem. We presented research agenda to investigate intriguing in web cluster systems: (i) providing proactive distribution, (ii) Analyzing user access patterns at web server side, (iii) applying adaptive prefetch rate in web cluster systems, (iv) estimating the available bandwidth in wireless networks for web streaming, and (v) releasing the corrupted web streaming data at web proxy server.

A. Web Switch Scheme

As the use of cluster systems increases, improving performance has been a critical issue. In this study, we propose a proactive request distribution scheme, called *ProRD*, and compare this with three other policies: WRR, LARD and Ext-LARD-PHTTP determine the policy that provides best results in terms of efficiency. WRR has a good load balancing capability, but its locality is so poor that it increases miss rates. In order to reduce the miss rates and improve secondary storage scalability, LARD can be used.

However, for large websites with immensely huge dataset, where caching considerable website content becomes impossible, performance of LARD degrades. Thus, we propose *ProRD* that employs ProActive locality-based and prefetch-aware request distribution which is complemented by prefetching at the backend servers. Such dynamic reconfiguration of the mining usage data in the web server's cache becomes a significant factor for the contribution of the performance the system. The simulation results with original website logs indicate that our system provides considerable improvement in the performance of the system.

B. Web Prefetch Scheme

Rapid growth in the number of web users and current HTTP frameworks makes it difficult to improve the performance through a prefetch scheme. Also, the access pattern of a web server system is not easily predictable because of the web cache mechanism and web object configuration. For proper web prefetching in a cluster environment, we introduce the *DPS* scheme to obtain the relationship information of the objects and increase the hit rate of the prefetched data. Also, we propose the *ARC* scheme to perform an efficient management of prefetch memory in cluster environments. Finally, we suggest the *MARD* to distribute the web workload to improve the efficiency in web prefetch.

For evaluation, we implement the prototype of web prefetch engine using PAPI and Apache web server in Linux. Also, we perform the simulation for verifying the benefit of our scheme in cluster environments. Experimental results show that our prefetch scheme improves the performance of web cluster systems up to 40% in various web workloads. There are three reasons for the improved performance of our scheme. i) Although our prefetch scheme loses the chance to increase the hit rate on the prefetch memory, it avoids the memory saturation and performance degradation caused by an excessive prefetching. ii) our prefetch scheme provides the adaptive prefetch rate at run time to maximize the prefetch benefit. iii) our prefetch scheme is adopted to the modern web framework and workloads.

C. Web Streaming Scheme

The internet and wireless home networks have undergone rapid growth which has led to an increase in streaming services through web system. This increase has necessitated attention to quality of service issues to web clients. The most challenging aspect of dynamic multimedia streaming service is the adaptive bit rate of each multimedia stream according to the network status; therefore, in this study, we focus on a method to estimate the available bandwidth of a wireless link. The method must have the following characteristics: (a) it should be applicable to real-time applications such as multimedia streaming services; (b) be simple and effective in estimating the available bandwidth, and (c) incur low overhead.

We present a new bandwidth estimation method, *IdleGap*, which can efficiently calculate the available bandwidth using the information collected from one node in a wireless network. Our *IdleGap* is simple and does not incur extra network overhead. The simulation result shows that our *IdleGap* outperforms the other probing and bandwidth estimation methods such as the ProbeGap and Spruce.

In addition, we show how our newly presented *ActiveSTB* improved the quality of the streamed media to client using early dropping and bandwidth estimation. We design an *ActiveSTB* to consider the challenges of effective usage of the shared wireless channel, reduction of the latency, and improvement of the QoS. Results demonstrate our *ActiveSTB* overcoming these issues by its use of extracting layer info from buffered stream data using early dropping and bandwidth estimation. Based on results, we believe the methods implemented in our *ActiveSTB* module will greatly enhance the quality of data streamed to clients, thus contributing to increased wireless home network usage and an increase in the growth of the STB market.

REFERENCES

[1] C. Huitema, "Network vs. server issues in end-to-end performance," Keynote address presented at the Performance and Architecture of Web Servers Workshop, Santa Clara, CA, 2000.

[2] E. V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving disk energy in network servers," in *Proc. of the International Conference on Supercomputing*, San Francisico, CA, 2003, pp. 86–97.

[3] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-aware request distribution in cluster-based network servers," in *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, 1998, pp. 205–216.

[4] R. Sarukkai, "Link prediction and path analysis using Markov chains," *Computer Networks*, vol. 33, pp. 377–386, June 2000.

[5] X.Dongshan and S. Junyi, "A new Markov model for web access prediction," *Computing in Science and Engineering*, vol. 4, pp. 34–39, November 2002.

[6] C. Bouras, A. Konidaris, and D. Kostoulas, "Predictive prefetching on the web and its potential impact in the wide area," *World Wide Web: Internet and Web Information System*, vol. 7, pp. 143–179, June 2003.

[7] J. Domenech, J. Sahuquillo, J. A. Gil, and A. Pont, "The impact of the web prefetching architecture on the limits of reducing user's perceived latency," in *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, Hong Kong, China, 2006, pp. 740–744.

[8] R. L. Carter and M. E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Technical Report TR-96-007, Computer Science Department, Boston University, Boston, MA, 1996.

[9] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls in end-to-end available bandwidth estimation," in *Proc. of ACM Internet Measurement Conference*, Taormina, Sicily, Italy, 2004, pp. 272–277.

[10] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 879–974, August 2003.

[11] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. of Internet Measurement Conference*, Miami Beach, FL, 2003, pp. 39–44.

[12] M.C. Rosu and D. Rosu, "Exploiting in-kernel data paths to improve I/O throughput and CPU availability," in *Proc. of the Winter 1993 USENIX Conference*, San Diego, CA, 1993, pp. 327–334.

[13] A. Cohen, S. Rangarajan, and H. Slye, "On the performance of TCP splicing for URL-aware redirection," in *Proc. of USENIX Symposium on Internet Technologies and Systems*, Boulder, CO, 1999, pp. 11–20.

[14] K. Rajamani and C. Lefurgy, "On evaluating request-distribution schemes for saving energy in server clusters," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, 2003, pp. 111–122.

[15] C. Li and K. Shen, "Managing prefetch memory for data-intensive online servers," in *Proc. of USENIX Conference on File and Storage Technologies*,

San Francisco, CA, 2005, pp. 19–23.

[16] B. S. Gill and L. A. D. Bathen, “Optimal multistream sequential prefetching in a shared cache,” *ACM Transactions on Storage*, vol. 3, Article No. 10, October 2007.

[17] C. G. Quinones, C. Madriles, J. Sanchez, P. Marcuello, A. Gonzalez, and D. M. Tullsen, “Mitosis compiler: An infrastructure for speculative treading based on pre-computation slices,” *ACM SIGPLAN Notices*, vol. 40, pp. 269–279, June 2005.

[18] Z. Zhang, X. Ma K. Lee, and Y. Zhou, “PFC: Transparent optimization of existing prefetching strategies for multi-level storage systems,” in *Proc. of International Conference on Distributed Computing Systems*, Beijing, China, 2008, pp. 740–751.

[19] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D.D.E. Long, “Managing flash crowds on the internet,” in *Proc. of International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Orlando, FL, December 2003, pp. 246–249.

[20] J. Domenech, A. Pont, J. Sahuquillo, and J. A. Gil, “A user-focused evaluation of web prefetching algorithms,” *Computer Communications*, vol. 30, pp. 2213–2224, July 2007.

[21] R. Kokku, P. Yalagandula, A. Venkataramani, and M. Dahlin, “NPS: A non-interfering deployable web prefetching system,” in *Proc. of USENIX Symposium on Internet Technologies and Systems*, Seattle, WA, 2003, pp. 183–196.

[22] A. E. Papathanasiou and M. L. Scott, “Aggressive prefetching: An idea whose

time has come," in *Proc. of Hot Topics in Operating Systems*, Santa Fe, NM, 2005, pp. 6–11.

[23] S. Liang, S. Jiang, and X. Zhang, "Step: Sequentiality and thrashing detection based prefetching to improve performance of networked storage servers," in *Proc. of International Conference on Distributed Computing Systems*, Toronto, Canada, 2007, pp. 64–73.

[24] T. Henderson, D. Kotz, and I. Abyzov, "The challenging usage of a mature campus-wide wireless network," in *Proc. of ACM Mobile Computing and Networking*, Philadelphia, PA, 2004, pp. 187–201.

[25] J. Shin, J. Kim, and C. C. J. Kuo, "Quality-of-Service mapping mechanism for packet video in differentiated service network," *IEEE Transaction on Multimedia*, vol. 3, no. 2, pp. 219–231, 2001.

[26] D. Quaglia and J. C. de Martin, "Delivery of MPEG video streams with constant perceptual quality of service," in *Proc. of International Conference Multimedia and Exhibition*, Politecnico di Torino, Italy, 2002, pp. 85–88.

[27] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proc. of Passive and Active Measurements*, Fort Collins, CO, 2002, pp. 14–25.

[28] K. Lakshiminarayanan, V. N. Padmanabhan, and J. Padhye, "Bandwidth estimation in broadband access networks," in *Proc. of Internet Measurement Conference*, Taormina, Sicily, Italy, 2004, pp. 314–321.

[29] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel, "Scalable content-aware request distribution in cluster-based network servers," in *Proc. of USENIX 2000*

Annual Technical Conference, San Diego, CA, 2000, pp. 11–25.

[30] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Dynamic cluster reconfiguration for power and performance," in *Compilers and Operating Systems for Low Power*, L. Benini. M. Kandemir and J.Ramanujam, Eds., chapter V, pp. 75–93. Kluwer Academic Publishers, Norwell, MA, 2003.

[31] Microsoft Corporation, *Installation and Performance Tuning of Microsoft Scalable Web Cache 3.0 Manual*, Microsoft Corporation, 2000, Available http://www.microsoft.com/technet.

[32] M. Bar, "Kernel Korner: kHTTPd, a kernel-based web server," *Linux Journal*, vol. 2000, Article No. 21, August 2000.

[33] P. Joubert, R. King, R. Neves, M. Russinovich, and J. Tracey, "High-performance memory-based web servers: Kernel and user-space performance," in *Proc. of USENIX Technical Conference*, Monterey, CA, June 2001, pp. 175–187.

[34] J. Bucy and G. Ganger, "The DiskSim simulation environment Version 3.0 Manual," Technical Report CMU-CS-03-102, Carnegie Mellon University, Pittsburgh, PA, 2003.

[35] Apache Software Foundation, *Apache Manual : Web Caching Guide*, Apache Software Foundation, 2009, Available http://httpd.apache.org/docs/2.2/caching.html.

[36] H. D. Schwetman, "Introduction to process-oriented simulation and CSIM," in *Proc. of Winter Simulation Conference*, New Orleans, LA, 1990, pp. 154–157.

[37] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *ACM SIGCOM Computer Communication Review*, vol. 26, pp. 22–36, July 1996.

[38] J. Borges and M. Levene, "Data mining of user navigation patterns," in *Proc. of the Workshop on Web Usage Analysis and User Profiling*, San Diego, CA, 1999, pp. 31–36.

[39] M. Deshpande and G. Karypis, "Selective Markov models for predicting web page accesses," *ACM Transactions on Internet Technology*, vol. 4, pp. 163–184, May 2004.

[40] B. D. Davison, "Learning web request patterns," in *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, M. Levine and A. Poulovassilis, Eds., chapter IX, pp. 435–460. Springer-Verlag, Berlin, Germany, 2004.

[41] T. Palpanas and A. Mendelzon, "Web prefetching using partial match prediction," in *Proc. of International Web Caching Workshop*, San Diego, CA, March 1999, pp. 6–26.

[42] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "A data mining algorithm for generalized web prefetching," *IEEE Transaction on Knowledge and Data Engineering*, vol. 15, pp. 1155–1169, September 2003.

[43] X. Chen and X. Zhang, "A popularity-based prediction model fore web prefetching," *IEEE Computer*, vol. 36, pp. 63–70, March 2003.

[44] Z. Ban, Z. Gu, and Y. Jin, "An online PPM prediction model for web prefetching," in *Proc. of Web Information and Data Management*, Lisbon, Portugal, 2007, pp. 89–96.

[45] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and adaptive web replication using content clustering," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 979–994, August 2003.

[46] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, pp. 101–106, January 2006.

[47] N.J. Tuah, M. Kumar, and S. Venkatesh, "Resource-aware speculative prefetching in wireless networks," *Wireless Networks*, vol. 9, pp. 61–72, January 2003.

[48] S. Drakatos, N. Pissinou, K. Makki, and C. Douligeris, "A context-aware prefetching strategy for mobile computing environments," in *Proc. of International Conference on Wireless Communications and Mobile Computing*, Vancouver, BC, Canada, 2006, pp. 1109–1116.

[49] P. Ferragina and A. Gulli, "A personalized search engine based on web snippet hierarchical clustering," in *Proc. of World Wide Web*, Chiba, Japan, 2005, pp. 801–810.

[50] D. Cheng, R. Kannan, S. Vempala, and G. Wang, "A divide-and-merge methodology for clustering," *ACM Transactions on Database Systems*, vol. 31, pp. 1499–1525, December 2006.

[51] A. Serbinski and A. Abhari, "Improving the delivery of multimedia embedded in web pages," in *Proc. of International Conference on Multimedia*, Augsburg, Germany, 2007, pp. 779–782.

[52] E. Meneses and O. Rodriguez-Rojas, "Using symbolic objects to cluster web documents," in *Proc. of World Wide Web*, Edinburgh, Scotland, 2006, pp. 967–968.

[53] D. Kim, N. Adam, V. Alturi, M. Bieber, and Y. Yesha, "A clickstream-based collaborative filtering personalization model: Towards a better performance," in *Proc. of ACM International Workshop on Web Information and Data Management*, Washington DC, 2004, pp. 88–95.

[54] L. Lu, M. Dunham, and Y. Meng, "Mining significant usage patterns from clickstream data," in *Proc. of WebKDD*, Chicago, IL, 2005, pp. 1–17.

[55] J. Zhu, J. Hong, and J. G. Hughes, "Using Markov models for web site link prediction," in *Proc. of ACM Conference on Hypertext and Hypermedia*, College Park, MD, 2002, pp. 169–170.

[56] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Model-based clustering and visualization of navigation patterns on a web site," *Data Mining and Knowledge Discovery*, vol. 7, pp. 399–424, October 2003.

[57] F. Khalil, J. Li, and H. Wang, "Integrating Markov model with clustering for predicting web page accesses," in *Proc. of Australasian World Wide Web*, Coffs Harbour, Australia, 2007, pp. 63–74.

[58] S. Rixner, "Memory controller optimization for web servers," in *Proc. of IEEE/ACM International Symposium on Microarchitecture*, Portland, OR, 2004, pp. 355–366.

[59] B. Gill and D. Modha, "SARC: Sequential prefetching in adaptive replacement cache," in *Proc. of USENIX Annual Technical Conference*, Anaheim, CA, 2005, pp. 293–308.

[60] Z. Wang, D. Burger, K. S. McKinley, S. K. Reinhardt, and C. C. Weems, "Guided region prefetching: A cooperative hardware/software approach," in

Proc. of International Symposium. on Computer Architecture, San Diego, CA, 2003, pp. 338–349.

[61] R. M. Rabbah, H. Sandanagobalance, M. Ekpanyapong, and W. Wong, "Compiler orchestrated prefetching via speculation and predication," in *Proc. of Architectural Support for Programming Languages and Operating Systems*, Boston, MA, 2004, pp. 189–198.

[62] J. Lu, A. Das, W. Hsu, K. Nguyen, and S. G. Abraham, "Dynamic helper threaded prefetching on the SUN UltraSPARC CMP processor," in *Proc. of IEEE/ACM International Symposium on Microarchitecture*, Barcelona, Spain, 2005, pp. 93–194.

[63] W. Zhang, B. Calder, and D. M. Tullsen, "An event-driven multithreaded dynamic optimization framework," in *Proc. of Parallel Architectures and Compilation Techniques*, St. Louis, MO, 2005, pp. 87–98.

[64] W. Zhang, D. M. Tullsen, and B. Calder, "Accelerating and adapting precomputation threads for efficient prefetching," in *Proc. of International Symposium on High Performance Computer Architecture*, Phoenix, AZ, 2007, pp. 85–95.

[65] G. Yadgar, M. Factor, and A. Schuster, "Karma: Know-it-all replacement for a multilevel cache," in *Proc. of USENIX Conference on File and Storage Technologies*, San Jose, CA, 2007, pp. 169–183.

[66] J. Dongarra, K. London, S. Moore, P. Mucci, D. Terpstra, H. You, and M. Zhou, "Experiences and lessons learned with a portable interface to hardware performance counters," in *Proc. of International Symposium on Parallel and Distributed Processing*, Nice, France, 2003, pp. 289–295.

[67] Standard Performance Evaluation Corporation, *SPECweb2005 Manual*, Standard Performance Evaluation Corporation (SPEC), Warrenton, VA, 2005.

[68] V. Ribeiro, M. Coates, R. Riedi, S. S. B. Hendricks, and R. Baraniuk, "Multifractal cross-traffic estimation," in *Proc. of ITC Special Seminar on IP Traffic Measurement, Modeling, and Management*, Monterey, CA, 2000, pp. (15–1)–(15–10).

[69] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proc. of Global Telecommunications Conference*, San Francisco, CA, 2000, pp. 415–420.

[70] A. Amamra, K. M. Hou, and J.P. Chanet, "Available bandwidth estimation in wireless ad hoc network: Accuracy and probing time," in *Proc. of International Conference on Computational Science and Engineering*, Sao Paulo, Brazil, 2008, pp. 379–387.

[71] A. Johnsson, M. Björkman, and B. Melander, "A study of dispersion-based measurement methods in IEEE 802.11 ad-hoc networks," in *Proc. of ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Las Vegas, NV, 2004, pp. 227–230.

[72] A. Amamra, K. M. Hou, and J.P. Chanet, "Evaluation of the performance of the SLoPS: Available bandwidth estimation technique in IEEE 802.11b wireless networks," in *Proc. of New Technologies, Mobility and Security*, Paris, France, 2007, pp. 123–132.

[73] M. Kim, N, N. Venkatasubramanian Dutt, and C. Talcott, "xtune: Online verifiable cross-layer adaptation for distributed real-time embedded systems," *ACM SIGBED Review*, vol. 5, Article No. 24, January 2008.

[74] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "Integrated power management for video streaming to mobile handheld devices," in *Proc. of ACM International Conference on Multimedia*, Berkeley, CA, 2003, pp. 582–591.

[75] S. Mohapatra, R. Cornea, H. Oh, K. Lee, M. Kim, N. Dutt, R. Gupta, A. Nicolau, S. Shukla, and N. Venkatasubramanian, "A cross-layer approach for power-performance optimization in distributed mobile systems," in *Proc. of IEEE International Parallel and Distributed Processing Symposium*, Denver, CO, 2005, pp. 218–226.

[76] W. Yuan and K. Nahrstedt, "Practical voltage scaling for mobile multimedia devices," in *Proc. of ACM International Conference on Multimedia*, New York, NY, 2004, pp. 924–931.

[77] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in *Proc. of ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003, pp. 149–163.

[78] M. C. Vuran and I. F. Akyildiz, "Cross-layer analysis of error control in wireless sensor networks," in *Proc. of IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, Washington, DC, 2006, pp. 585–594.

[79] M. V. D. Schaar and D. S. Turaga, "Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission," *IEEE Transactions on Multimedia*, vol. 9, pp. 185–197, January 2007.

[80] I. V. Bajic, "Efficient cross-layer error control for wireless video multicast," *IEEE Transactions on Broadcasting*, vol. 53, pp. 276–285, March 2007.

[81] K. Lee, A. Shrivastava, M. Kim, N. Dutt, and N. Venkatasubramanian, "Mitigating the impact of hardware defects on multimedia applications: A cross-layer approach," in *Proc. of ACM International Conference on Multimedia*, Washington, DC, 2008, pp. 319–328.

[82] B. Shen, S. Lee, and S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks," *IEEE Transaction on Multimedia, Special Issue on Streaming Media*, vol. 6, pp. 375–386, 2004.

[83] P. Schojer, L. Boszormenyi, H. Hellwagner, B. Penx, and S. Podlipnig, "Architecture of a quality based intelligent proxy (QBIX) for MPEG-4 videos," in *Proc. of World Wide Web*, Budapest, Hungary, 2003, pp. 394–402.

[84] S. Chan, C. W. Kok, and A. K. Wong, "Multimedia streaming gateway with jitter detection," in *Proc. of International Conference on Communication*, Anchorage, AK, May 2003, pp. 1875–1879.

[85] T. Hsu C. Huang and C. Chang, "A proxy-based adaptive flow control scheme for media streaming," in *Proc. of ACM Symposium on Applied Computing*, Madrid, Spain, May 2002, pp. 750–754.

[86] L. Gae, Z. Zhang, and D. F. Towsley, "Proxy-assisted techniques for delivering continuous multimedia streams," *IEEE/ACM Transactions on Network*, vol. 11, pp. 884–894, December 2003.

[87] S. Chan, C. W. Kok, and A. K. Wong, "Multimedia streaming gateway with jitter detection," *IEEE Transactions on Multimedia*, vol. 7, pp. 585–592, 2005.

[88] A. Laursen, J. Olkin, and M. Porter, "Oracle media server: Providing consumer based interactive access to multimedia data," *ACM SIG on Management of*

Data, vol. 23, pp. 470–477, June 1994.

[89] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, pp. 1103–1120, September 2007.

[90] M. Davis, "A wireless traffic probe for radio resource management and QoS provisioning in IEEE 802.11 WLANs," in *Proc. of ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Venice, Italy, 2004, pp. 234–243.

[91] LAN MAN Standards Committee, "Wireless lan medium access control (MAC) and physical layer (PHY) specifications high-speed physical layer in the 5 ghz band," Technical Standard IEEE Std 802.11a, IEEE Standards Association, Piscataway, NJ, October 1999.

[92] LAN MAN Standards Committee, "Wireless lan medium access control (MAC) and physical layer (PHY) specifications high-speed physical layer in the 2.4 ghz band," Technical Standard IEEE Std 802.11b, IEEE Standards Association, Piscataway, NJ, October 1999.

[93] L. Gae, Z. Zhang, and D. F. Towsley, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proc. of ACM Symposium on Applied Computing*, Rome, Italy, 2001, pp. 236–251.

[94] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," in *Proc. of ACM MobiCom*, Atlanta, GA, 2002, pp. 25–35.

[95] A. Kamerman and L. Monteban, "WaveLAN II: A high-performance wireless

LAN for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118–133, August 1997.

[96] MacGillivray Freeman, "Amazing Caves," Laguna Beach, CA, 2000, MacGillivray Freeman Films.

[97] Paul Greengrass, "Bourne Ultimatum," Los Angels, CA, 2007, Universal Pictures.

[98] Francis Lawrence, "I Am Legend," Burbank, CA, 2007, Warner Bros. Pictures.

[99] David Silverman, "Simpsons," Los Angels, CA, 2007, 20th Century Fox Film Corporation.

[100] Raymond Martino, "To The Limit," Los Angels, CA, 1995, PM Entertainment Group.

[101] Joint Video Team (ITU-T and ISO/IEC), *Joint Scalable Video Model (JSVM) Software Manual*, Joint Video Team (ITU-T and ISO/IEC), 2007, Available http://lontra.org/pub/video/jsvm/current/jsvm/SoftwareManual.doc.

APPENDIX A

WEB PREFETCH SCHEME

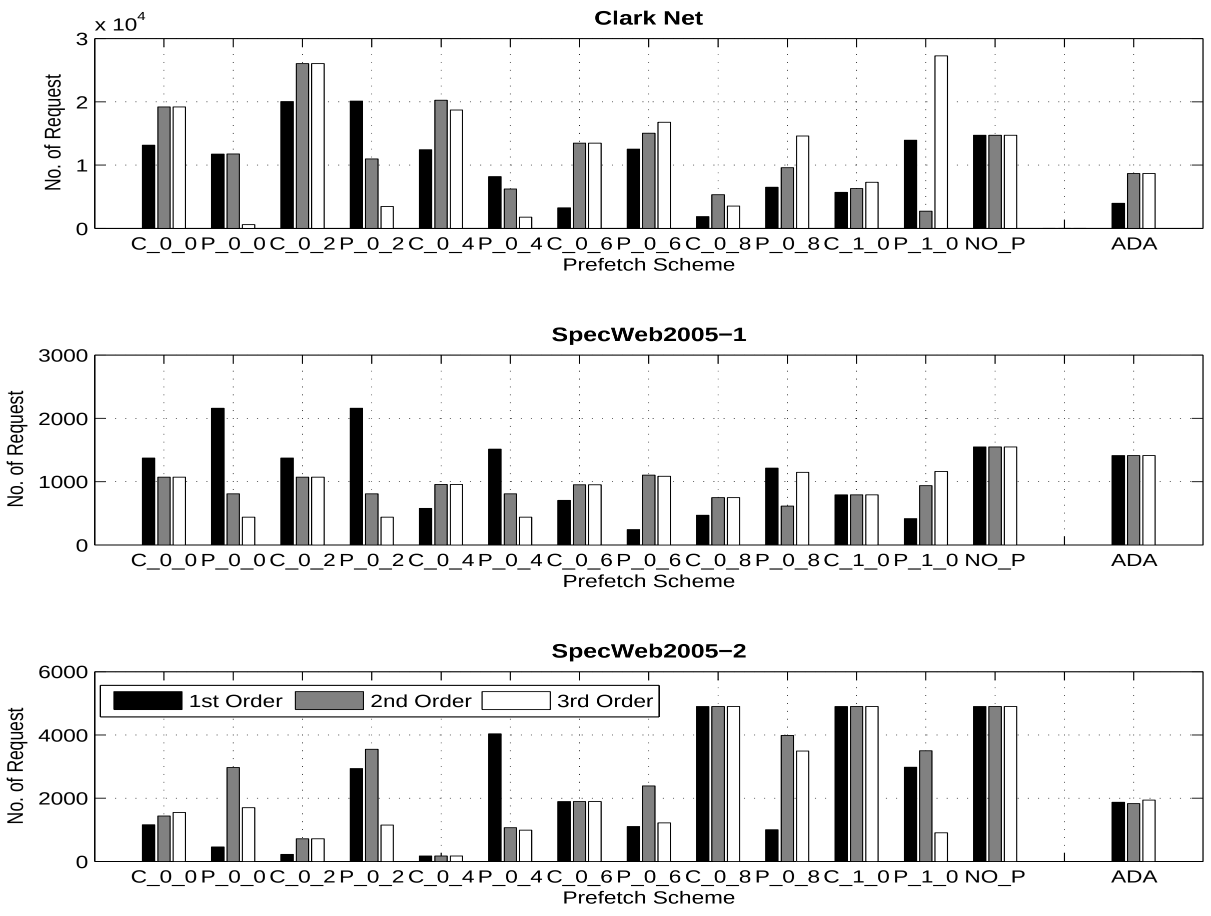

Fig. 34. Standard Deviation of Distribution (ClarkNet, SpecWeb2005-1, SpecWeb2005-2)

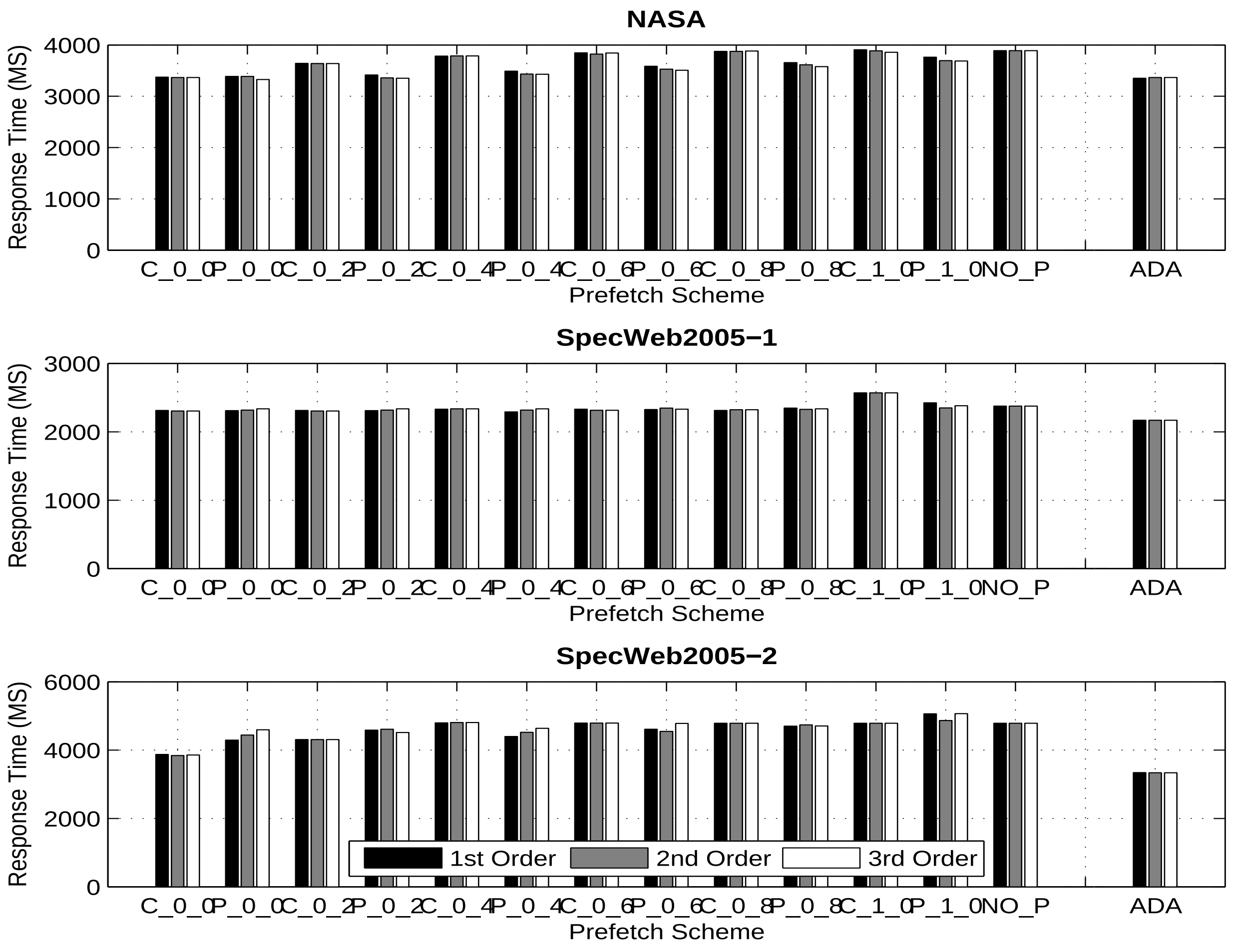

Fig. 35. Web Response Time in Prefetch Schemes (NASA, SpecWeb2005-1, SpecWeb2005-2)

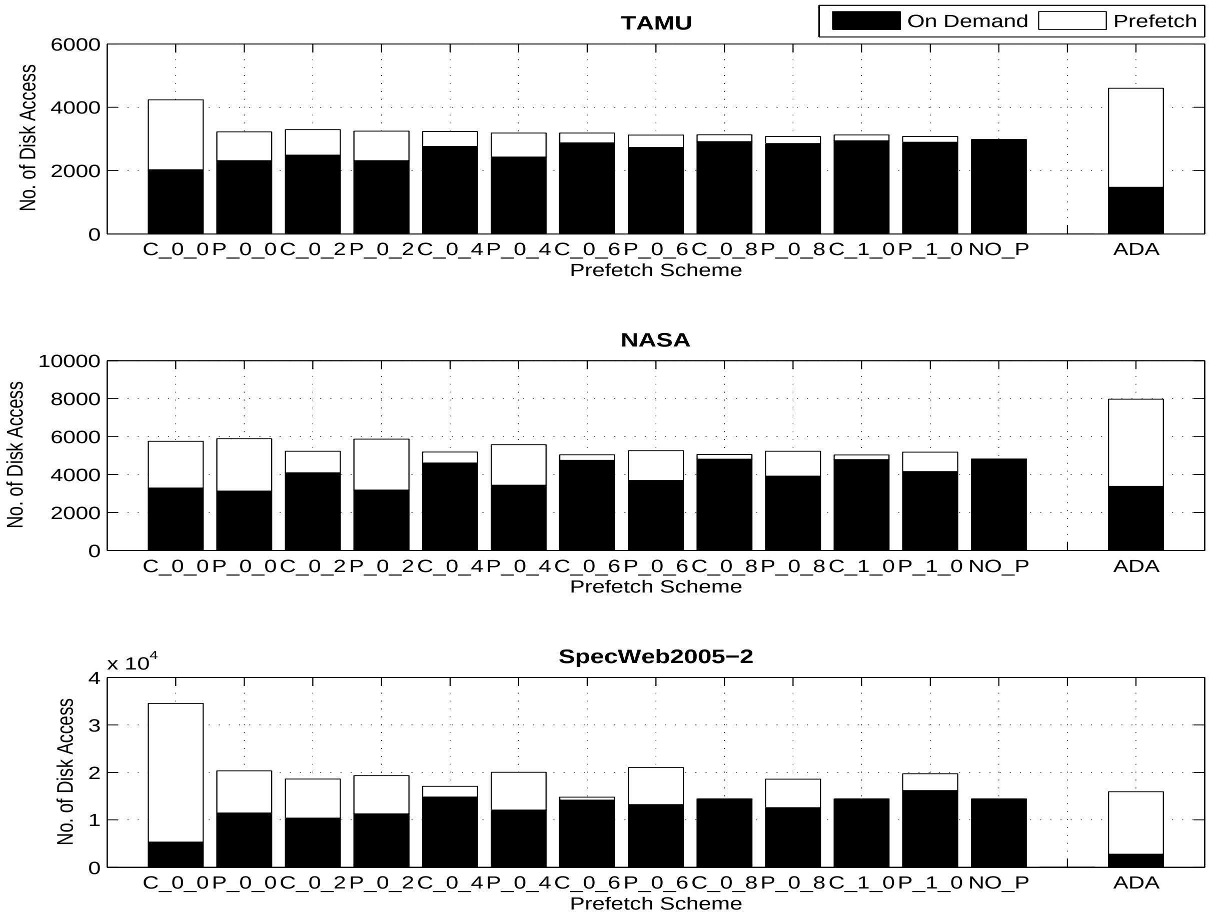

Fig. 36. Number of Disk Access (NASA, ClarkNet, SpecWeb2005-1)

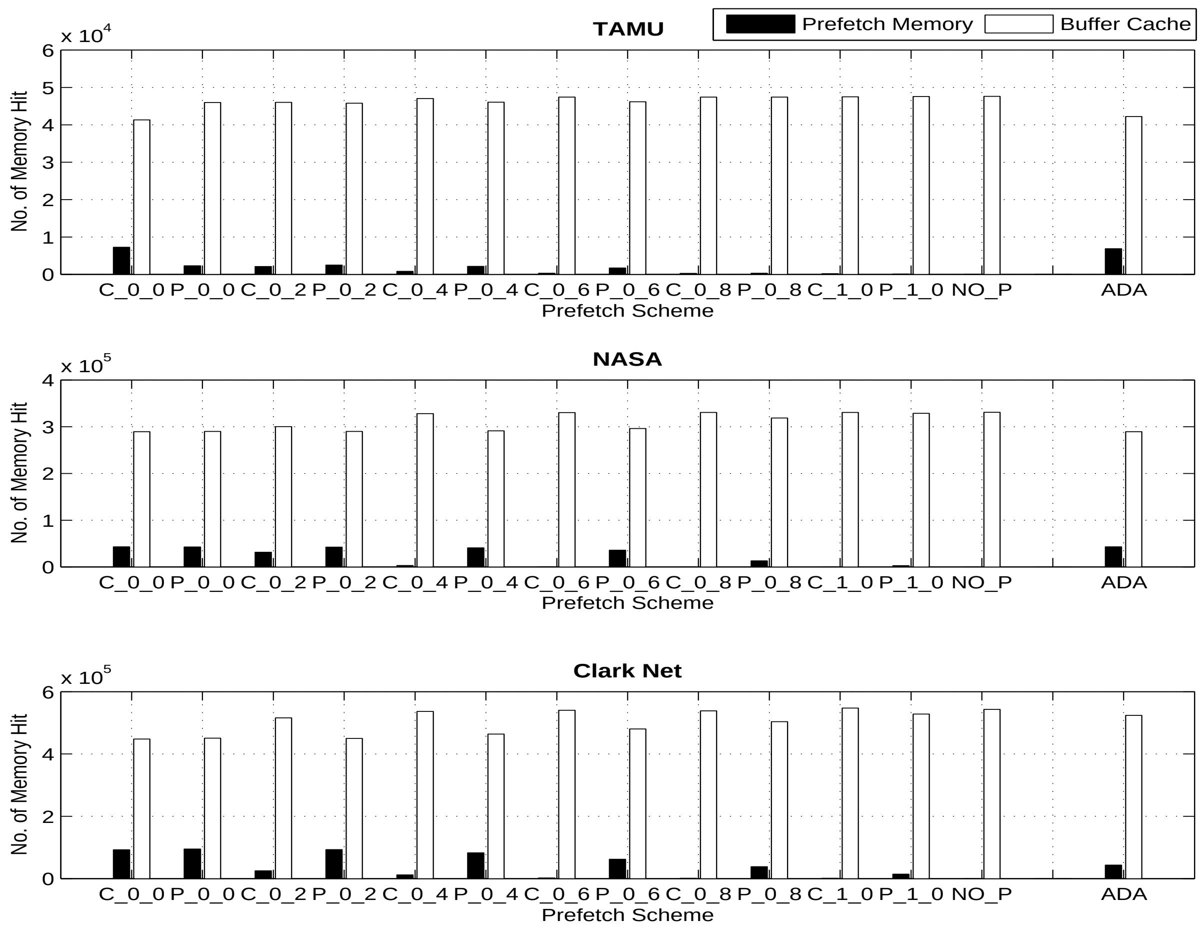

Fig. 37. Number of Hits on Memory (TAMU, NASA, ClarkNet)

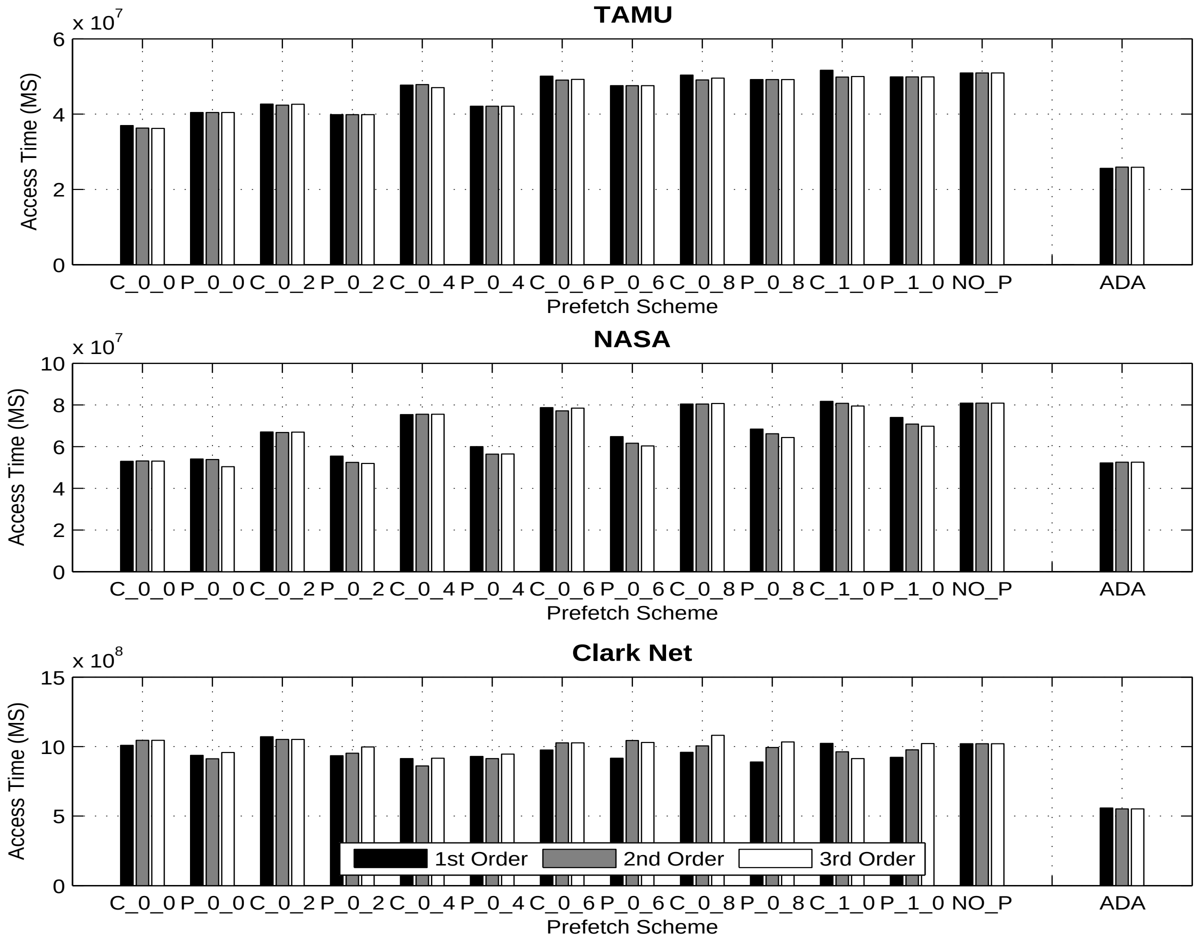

Fig. 38. Disk Access Time on the Demanding Requests (TAMU, NASA, ClarkNet)

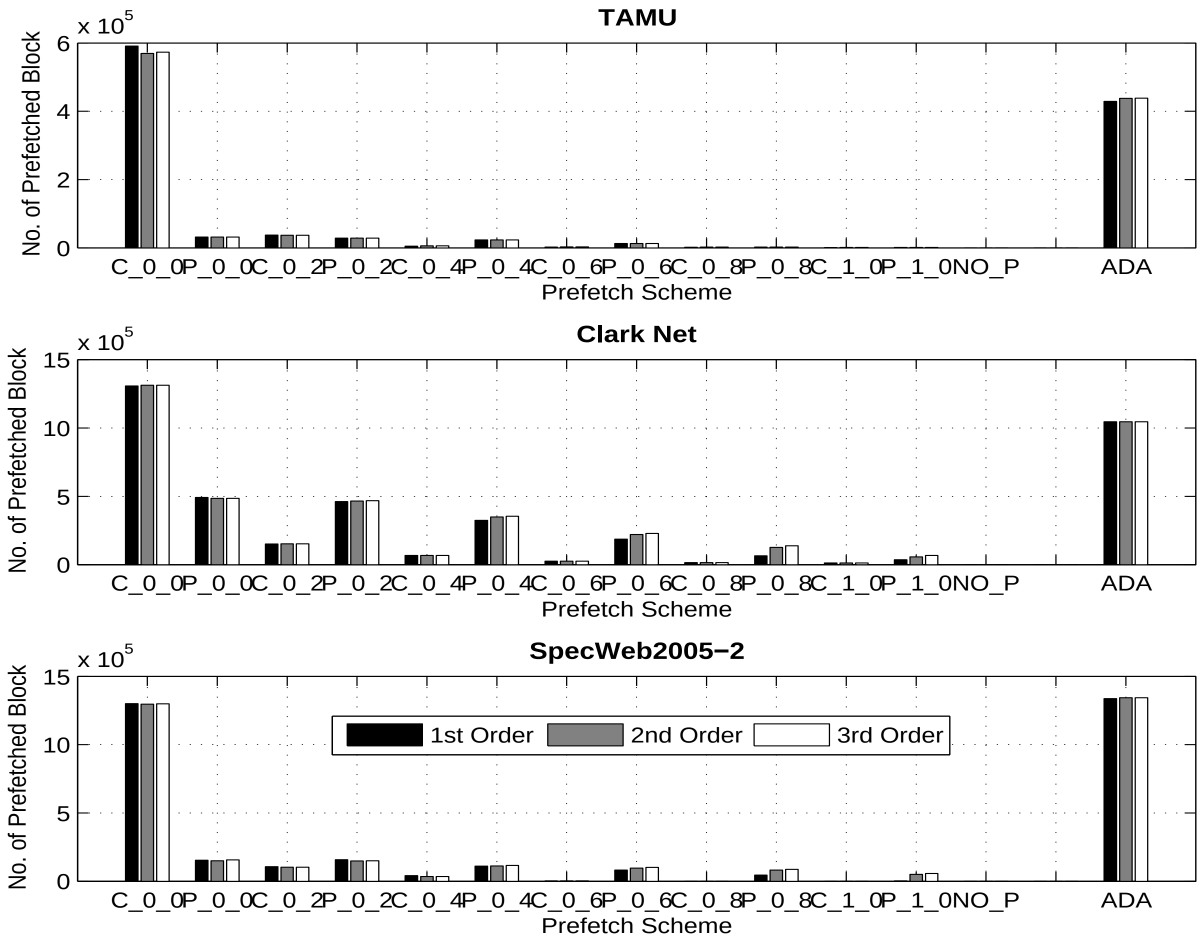

Fig. 39. Prefetched Blocks (TAMU, ClarkNet, SpecWeb2005-2)

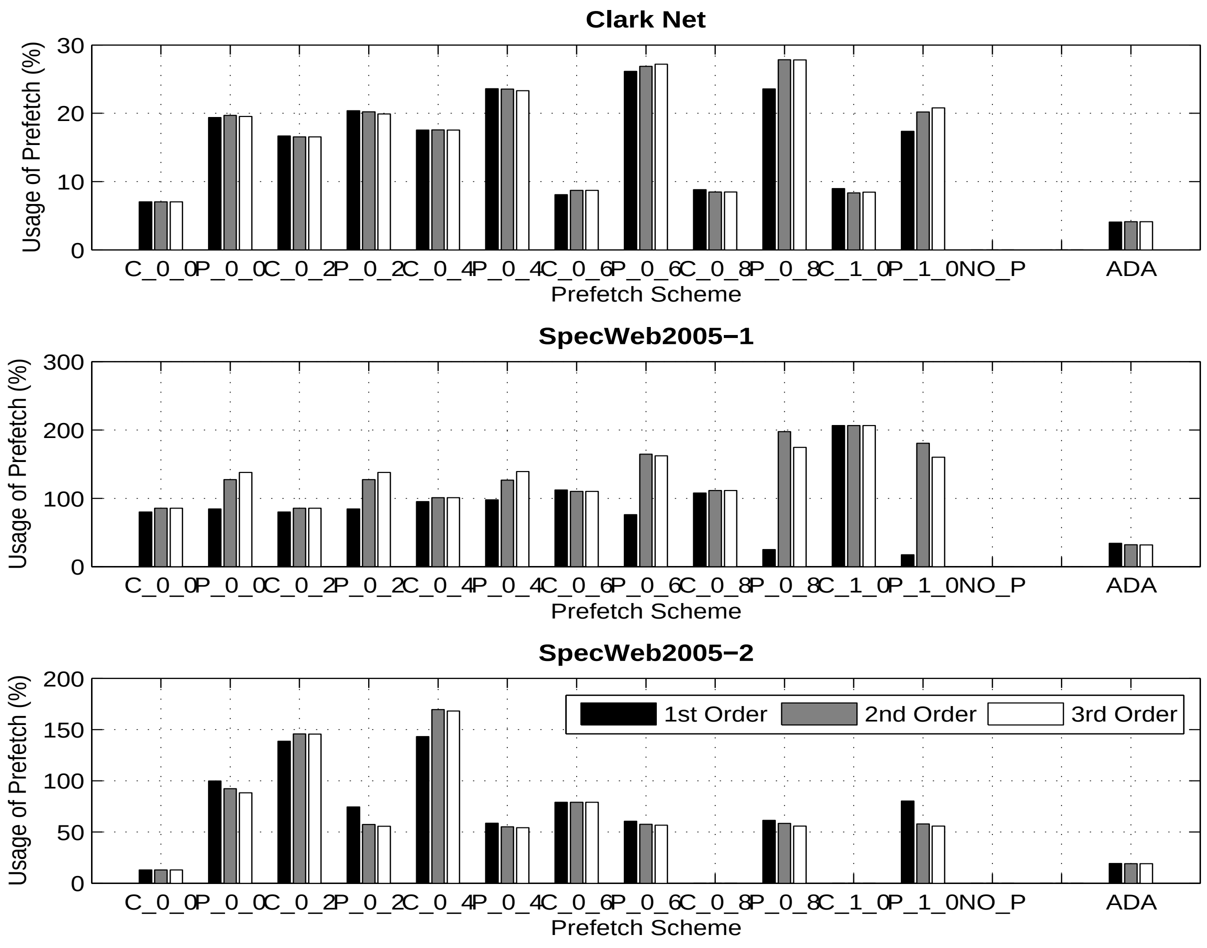

Fig. 40. Usage of Prefetched Blocks (ClarkNet, SpecWeb2005-1, SpecWeb2005-2)

APPENDIX B

ADAPTIVE STREAMING SERVICE

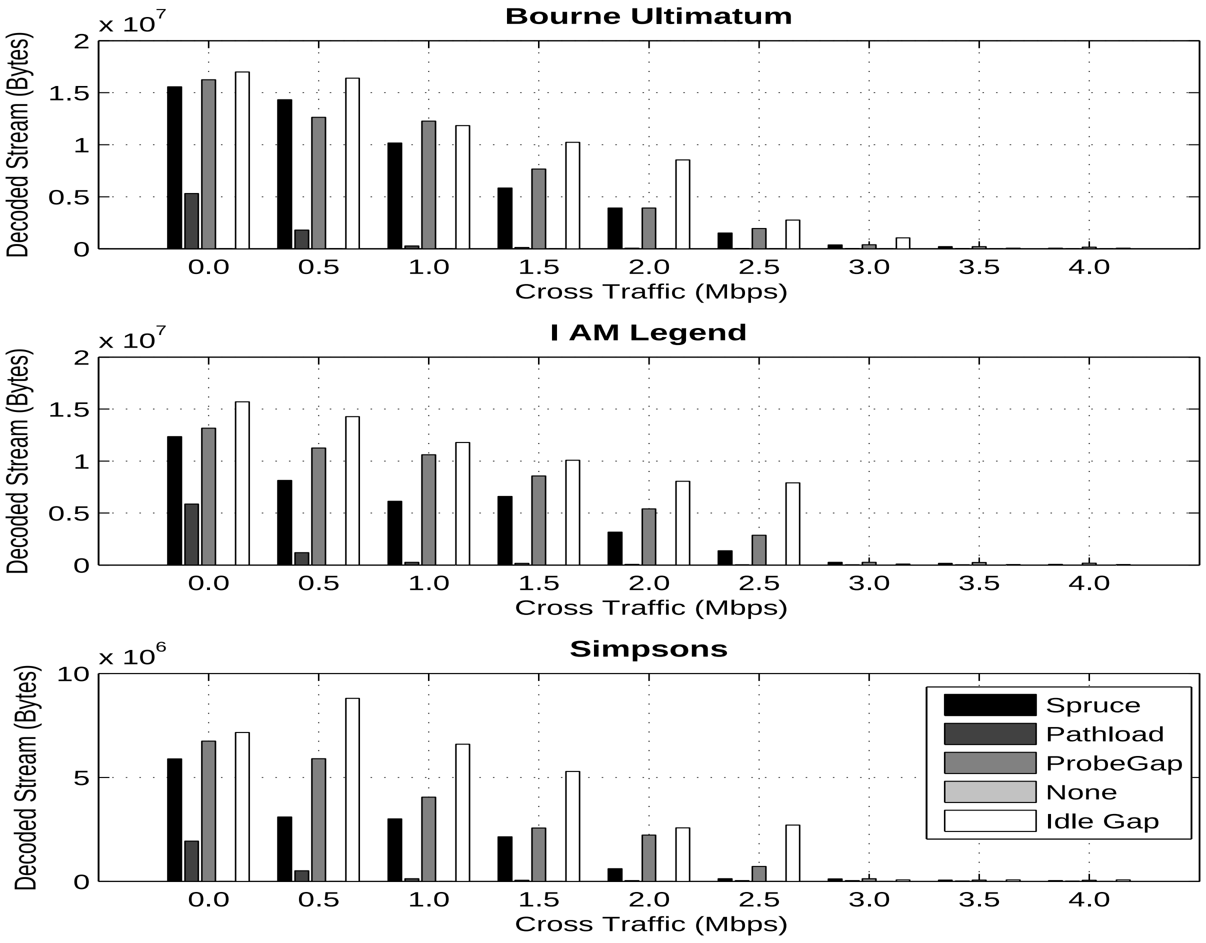

Fig. 41. The Size of Decoded Streams (*Bourne Ultimatum*, *I Am Legend*, *Simpsons*)

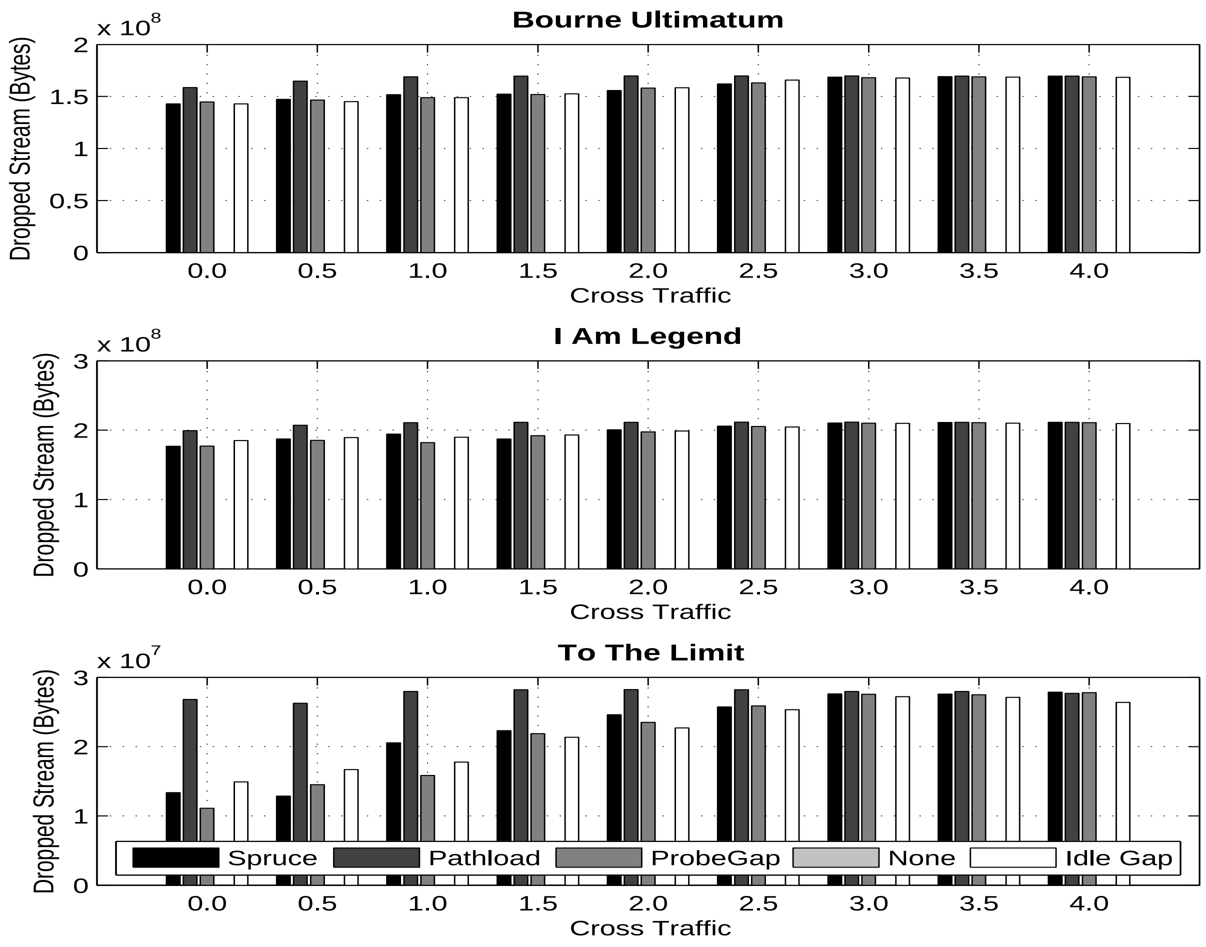

Fig. 42. The Early Dropped Stream (*Bourne Ultimatum*, *I Am Legend*, *To The Limit*)

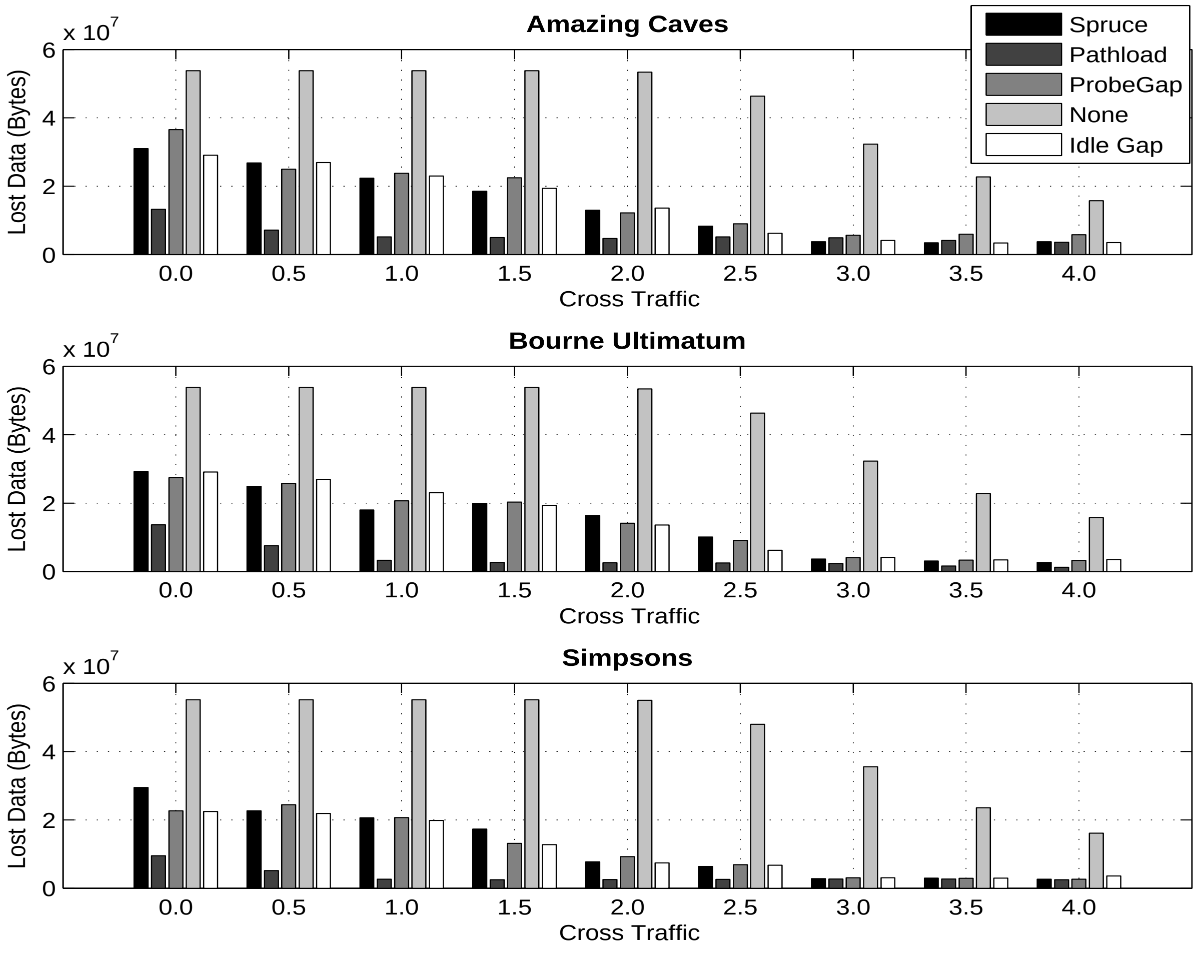

Fig. 43. The Indirect Loss (*Amazing Caves*, *Bourne Ultimatum*, *Simpsons*)

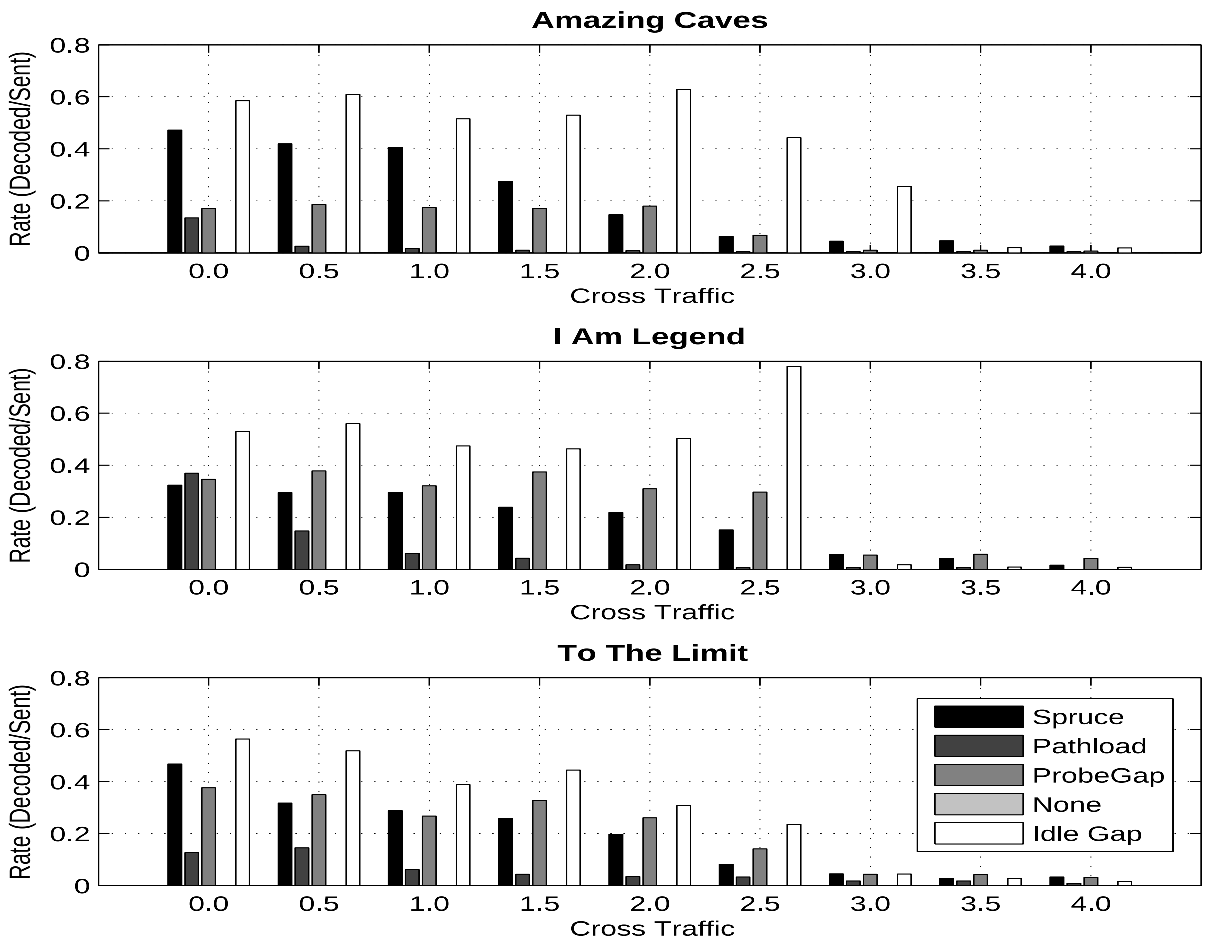

Fig. 44. The Ratio between Sent Size and Decoded Size (*Amazing Caves*, *I Am Legend*, *To The Limit*)

APPENDIX C

STARTUP DELAY

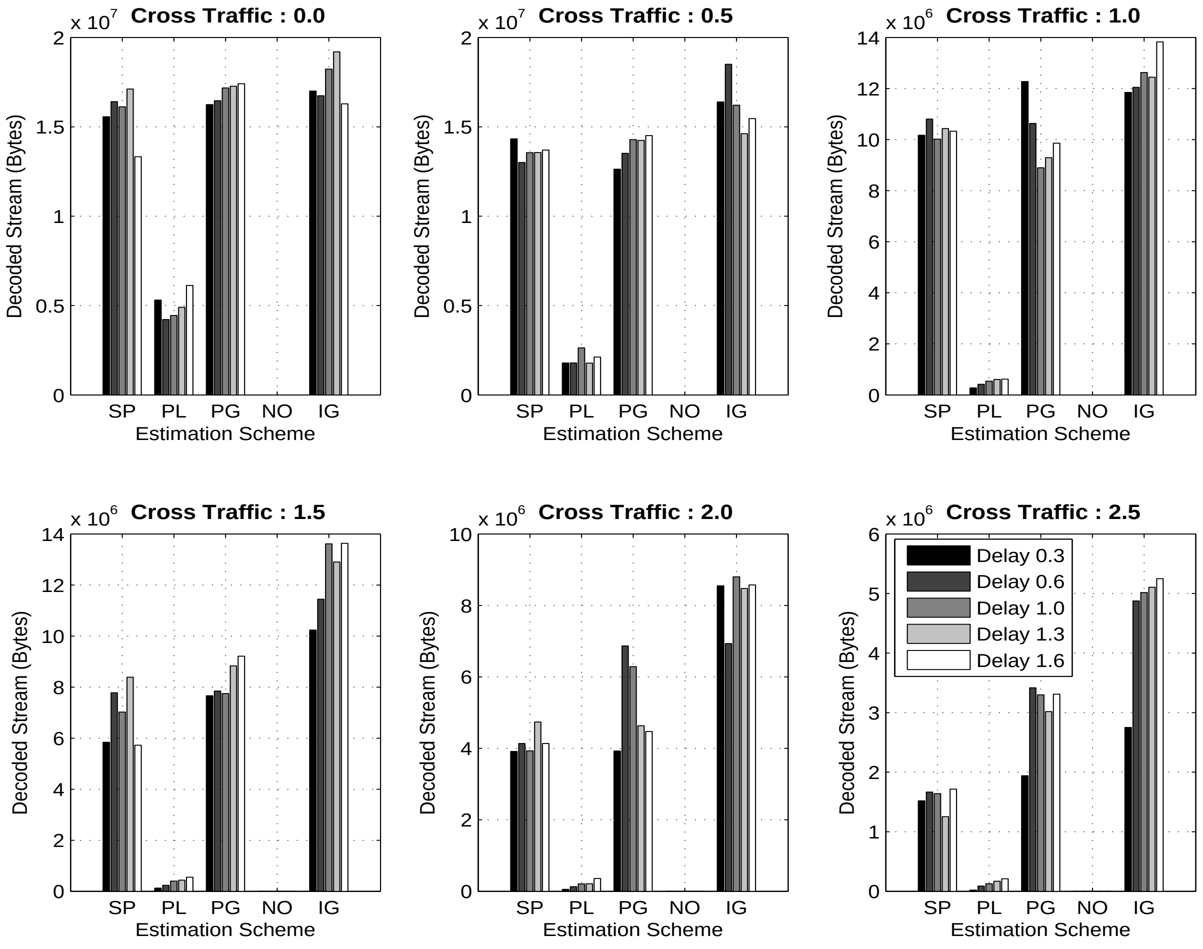

Fig. 45. The Size of the Decoded Stream of the *Amazing_Caves* under Various Startup Delay

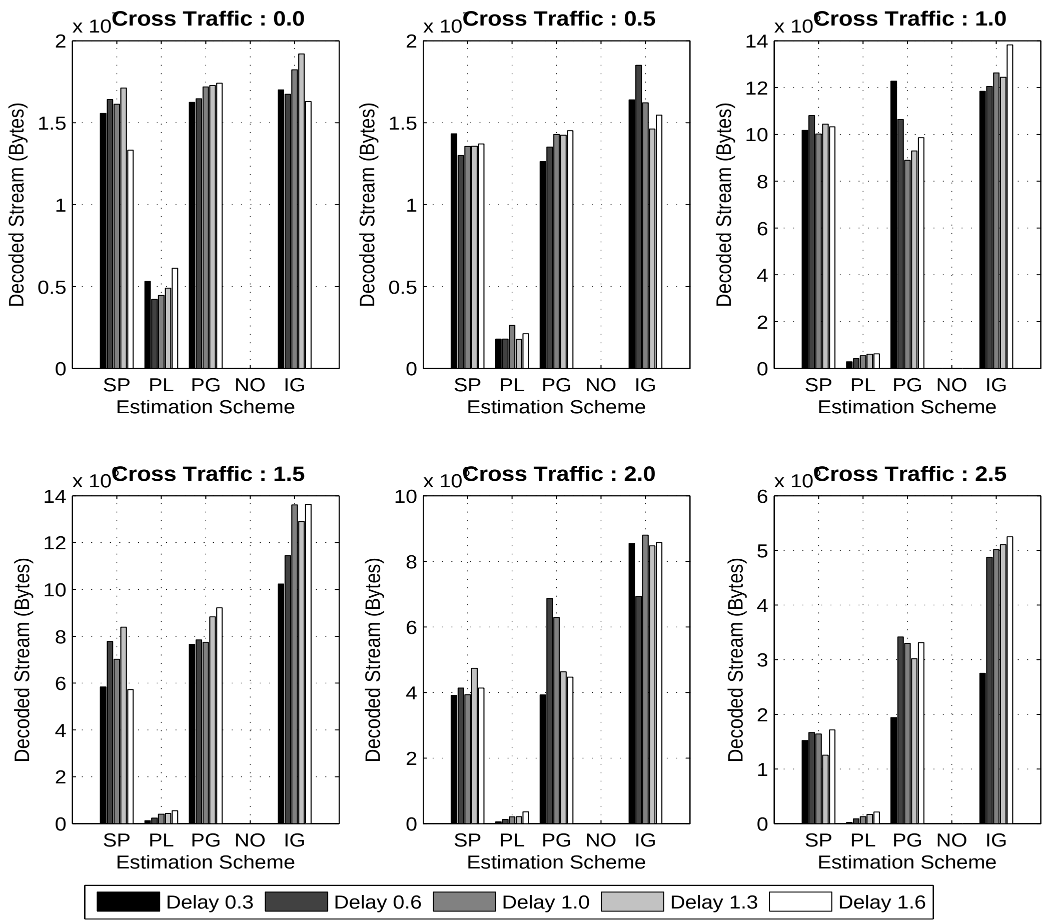

Fig. 46. The Size of the Decoded Stream of the *Bourne Ultimatum* under Various Startup Delay

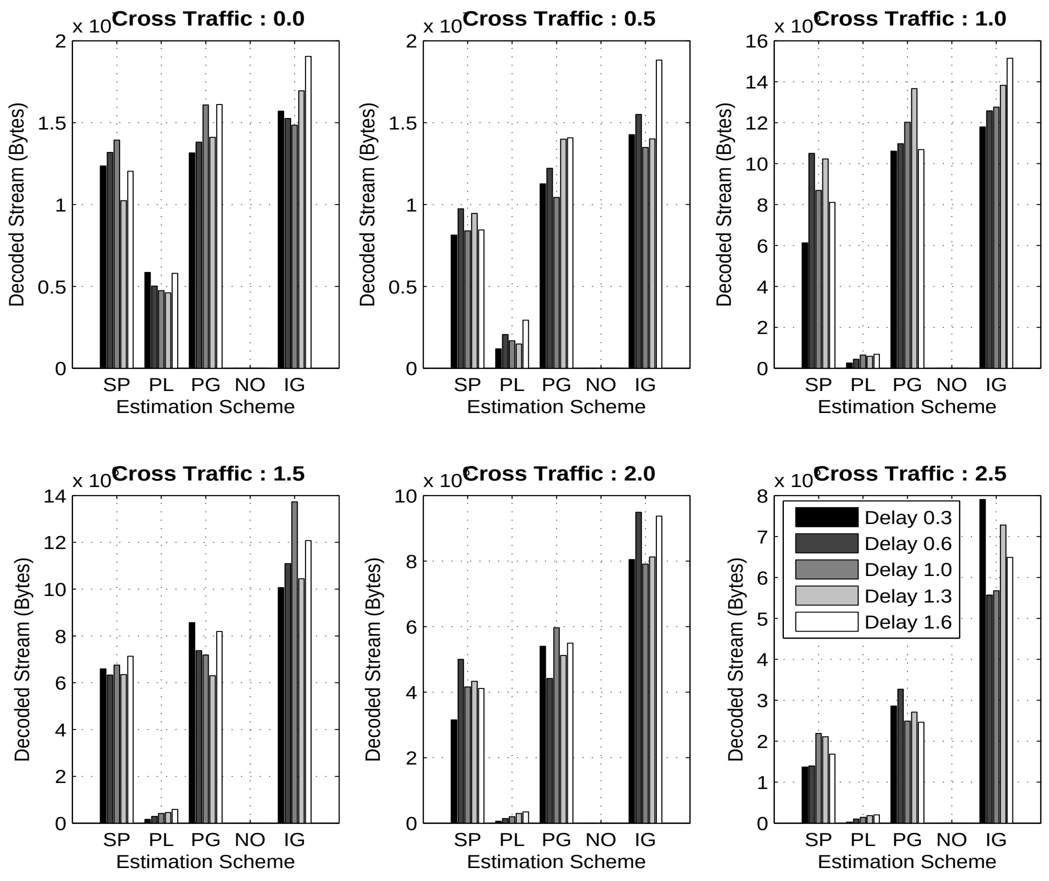

Fig. 47. The Size of the Decoded Stream of the *I Am Legend* under Various Startup Delay

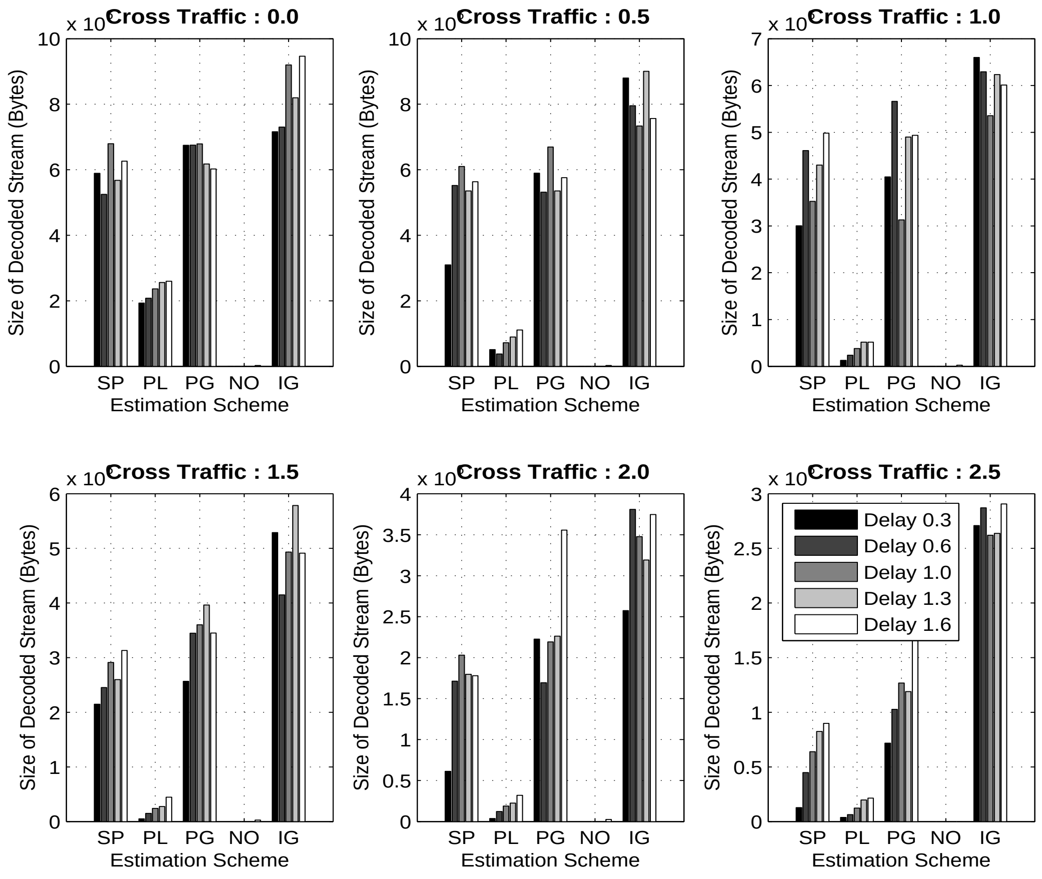

Fig. 48. The Size of the Decoded Stream of the *Simpsons* under Various Startup Delay

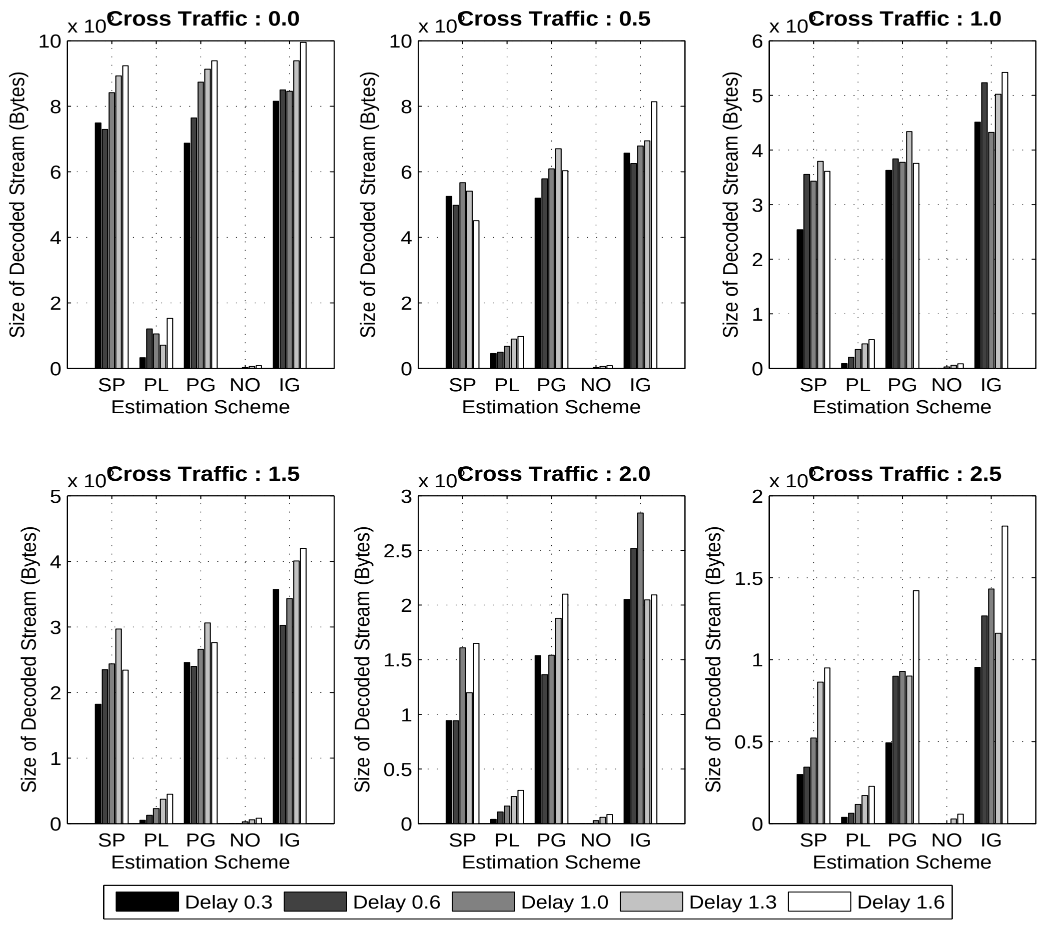

Fig. 49. The Size of the Decoded Stream of the *To The Limit* under Various Startup Delay

VITA

Heung Ki Lee

301 Harvey R. Bright Building

College Station, TX 77843-3112

Email: hklee@cs.tamu.edu

EDUCATION

- Ph.D. Texas A&M University (09/2003-12/2009) College Station, TX (USA)
- M.S. Chungnam University (03/2000-08/2002) Daejeon, South Korea (ROK)
- B.S. Chungnam University (03/1993-02/2000) Daejeon, South Korea (ROK)

PUBLICATIONS

- H.K. Lee, B. S. An and E.J. Kim, "Adaptive Prefetching Scheme Using Web Log Mining in Cluster-based Web Systems," in *Proc. of International Conference on Web Services(ICWS)*, Los Angels, CA, 2009, pp. 903-910.
- H.K. Lee, V. Hall, K. H. Yum, K. I. Kim and E. J. Kim, "Bandwidth Estimation In Wireless LANs For Multimedia Streaming Services," in *Proc. of International Conference on Multimedia & Expo (ICME)*, Toronto, Canada, 2006, pp. 1181-1184.

This typist for this dissertation was Heung Ki Lee.