PH.D. THESIS

Market- Based Resource Allocation for Content Delivery in the Internet

by Ozgur Ercetin
Advisor: Leandros Tassiulas

CSHCN PhD 2002-4
(ISR PhD 2002-4)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

ABSTRACT

Title of Dissertation: MARKET-BASED RESOURCE
ALLOCATION FOR CONTENT DELIVERY
IN THE INTERNET

Özgür Erçetin, Doctor of Philosophy, 2002

Dissertation directed by: Professor Leandros Tassiulas
Department of Electrical and Computer Engineering

The Internet has evolved into a worldwide information backbone with vast user base. Internet users are diverse and they receive varying utilities from their network connections. In many cases high-level user decisions appear to have direct consequences on the network performance. In this thesis, our motivation is to characterize the effects of user behaviors in terms of lower layer network metrics such as network latency. We consider the Content Delivery Networks for our analysis, since they are the interconnection of network elements at the application layer and thus are directly affected by the user preferences.

It has been a common practice to use caches to store the most popular data in order to improve user latency and reduce the network load. Recently, a more systematic approach to the caching has been developed in the framework of Content Delivery Networks. Content Delivery Networks (CDNs) are the networks of caches that are distributed throughout the Internet serving user requests directed to their subscriber

web sites (publishers). They distribute the publisher's original content intelligently to the caching servers (surrogates) all over the world. Users receive their information from the surrogates, which are closer and usually much less loaded than the origin server. The objective is to minimize the user latency by intelligently distributing the content and serving the user requests from the most efficient surrogates. We use price-directed market based algorithms to achieve this goal. As it is the case in the current Internet, we model the agents with a selfish self-maximizing behavior and define the problem as a non-cooperative game played among the publishers and the surrogates. We show that the system has an equilibrium and if this equilibrium is unique, even though the agents are non-cooperative we achieve the global optimum. We also determine a uniqueness condition, which states that if the publishers are not willing to pay high amounts and the cache sizes are sufficiently small, then the equilibrium is unique. We noticed that the global system optimum that minimizes total average user latency requires the publishers to make very high investments, which in practice may prohibit the applicability of the distributed market method. Thus, we consider an investment strategy, which leads to a near-optimum system solution at much lower investments.

The abovementioned method gives publishers infinite granularity to determine their quality of service. Next, we investigated the case where the publishers can offer only finite number of QoS classes. In this model, surrogate partitions its total capacity among different service classes and among each class publishers receive equal shares of the resources. In our model, publishers try to get as large cache space as possible, while the surrogate is required to achieve fair allocation among the publishers. Specifically, each publisher should be charged the same if they receive equal share of caching space. We determine the optimal pricing strategy of the surrogate maximizing its revenue. We also analyzed the competition among two surrogates under this model and determined the condition that leads to a Nash equilibrium. We showed that at equilibrium surrogates peer as if they are a single combined surrogate server.

MARKET-BASED RESOURCE ALLOCATION FOR CONTENT DELIVERY IN THE INTERNET

by

Özgür Erçetin

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2002

Advisory Committee:

Professor Leandros Tassiulas, Chairman/Advisor
Professor Michael O. Ball
Professor John Baras
Professor Anthony Ephremides
Professor Sennur Ulukus

DEDICATION

To my parents, Aynur and Özcan

Sevgili aileme

ACKNOWLEDGEMENTS

And at last I am here. This is the part of my thesis that I really anticipated to write. Although it may be the one of the first things you see in my thesis, for me this section brings the end to my thesis. This is the part I can write fiction...

Like so many other PhD students, my doctoral study was full of hard work, sacrifice, sometimes despair and at the same time a lot fun (sounds not normal I know, but no PhD graduate is exactly normal!). I feel so fortunate and gifted to have this experience and finish it too...

I am very much indebted to my advisor Prof. Tassiulas for his continuous support, friendship and enlightenment not only during my PhD but also during my Masters studies. My work matured under his supervision and without his deep insight in the field and his invaluable directions I would not be able to accomplish this enormous task.

I would also like to thank to my committee members Prof. Ephremides, Prof. Baras, Prof. Ball and Prof. Ulukus for their valuable comments. My

special thanks goes to Prof. Ephremides, who was present both in my master's defense as well as proposal exam, and Prof. Baras who was in my proposal exam.

I was so fortunate to be surrounded by many great friends during my studies in College Park. They made this place home away from home for me and provided comfort at hard times and fun at the good times. I gained a lot from their friendship and our discussions, whether those discussions are on serious topics like research and politics or simply movies and cars.

Above all, I would like to thank to my family. I am so gifted to have such a wonderful family. They brought me to life, they raised me and sacrificed to let me go to follow my dreams. I thank them for their support, encouragement and unbounded love! Thank you very much mom, dad, brother and sister. I dedicate this thesis to you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

MARKET-BASED RESOURCE
ALLOCATION FOR CONTENT DELIVERY
IN THE INTERNET

Özgür Erçetin

July 1, 2002

This comment page is not part of the dissertation.

Typeset by LaTeX using the `dissertation` class by Pablo A. Straub, University of Maryland.

Chapter 1

Introduction and Background

Internet has gained an important place in our lives by enabling us to reach any information anytime and anywhere we want. From its early stages as an experimental network (ARPANET) connecting defense and academic institutions, Internet has grown exponentially in dimensionality and user traffic. This growth has resulted in congestion over the network links and excruciating end-user delays while downloading the information. Caches and proxies are installed to alleviate the congestion on the network links and the delays associated with them. Caches and proxies together store information closer to the users and serve the user requests on behalf of the corresponding servers. Although caches and proxies have been successful in alleviating the congestion, they are basically best-effort solutions to the more important problem of content distribution in the Internet. The objective of content distribution problem is to determine the best location between the original source and client for the content to be delivered. This problem has been investigated for the past couple of decades not only in the context of web content delivery but also in the context of file storage in large scale file systems. Recently, Content Delivery Networks (CDNs) have been developed to solve this problem.

In the following, we first review the past research in the area of content delivery,

especially in the context of web access. We also review the current efforts in integrating price-based user-centric network control. This topic has received much attention recently, since it promises to migrate the control of the network to the edges, and thus reducing the complexity of the network. Furthermore, by using certain incentives, the users can be differentiated and the most efficient resource allocation can be achieved.

In Chapter 2, we describe the *content distribution* and its more practical counterpart *content delivery* problems. Unlike the content distribution problem, the content delivery problem maximizes the total web server utility by efficiently distributing the content and routing the user requests to the most suitable caches in the network. We develop a distributed algorithm for efficient dissemination and routing by price-oriented market-based resource allocation schemes. In Chapters 3 and 4, we analyze the operation of the distribution and routing sub-problems. In Chapter 5, we analyze the possibility of best-effort QoS by nonlinear pricing mechanisms.

1.1 Internet Content Delivery

1.1.1 World Wide Web Overview

The ease of use of World Wide Web (WWW) applications such as Microsoft's Internet Explorer and Netscape has facilitated the revolution in the communications. Common users can now access the network and retrieve any type of information they wish. Users employ an agent that is called *browser*, which displays to the user the requested information and provides numerous configuration and navigational features. In WWW, the information is available in the form of a web page. A *web page* is a collection of web objects. An *object* is a file such as an HTML (Hyper Text Markup Language) file, a JPEG or GIF image, a Java applet, an audio clip, etc. The base

HTML page references the other objects in the page with the objects' Uniform Resouce Locator, or URL in short. The *URL* uniquely defines the location of an object in human-understandable format. URL consists of two parts: host name and the directory, and the name of the object in the host computer where the object resides. A *web server* stores the web objects, each addressable by an URL.

Hypertext Transfer Protocol (HTTP) is the communications protocol used between web applications. HTTP defines the structure of the messages and how these messages are exchanged between the servers and the users during the transfer of the web pages. HTTP is an application level protocol that use reliable TCP (Transport Control Protocol) as an underlying transport layer protocol. In order for an HTTP client to initiate a TCP connection with the server, the client requires the IP address of the server itself. In HTTP, the servers are assigned alphanumeric names for ease of use for the human users. These names are associated with IP addresses, where the objects pointed by the URLs reside. At the beginning of a HTTP connection, the user first determines the IP address of the server from the server's URL. This is done by Internet directory service called Domain Name Server (DNS). The details of the DNS is discussed in the later sections. TCP connection between the client and the server provides reliable data transfer between the two parties.

HTTP can use both non-persistent and persistent connections. With a non-persistent connection, each object referenced by the main HTML page is delivered to the user by a separate TCP connection. The server tears down the TCP connection after sending the requested object. However, these connections can be set up in parallel. With persistent connections, the server leaves the TCP connection open after sending the requests. Subsequent requests and responses between the same client and the server can be sent over the same connection. Consequently, the delay due to

three-way handshake for multiple TCP connection set up can be avoided.

1.1.2 Internet Caching

With explosive growth of demand for Internet services, the network traffic has increased exponentially. As the load exceeded the network capacity, network congestion and server overloading become common phenomena resulting in increased traffic delays. Web caches are developed and implemented to reduce the traffic congestion and the latency. Web caches are located between Web servers (or origin servers) and users, and satisfies HTTP requests on behalf of the servers. Web cache has its own storage space and keeps in this storage copies of the recently requested objects. In case another request for the same object is received, the request is immediately satisfied by the cache avoiding the transfer of document from the distant original server.

Caches reduce latency by responding to the user requests from a closer location compared to the original servers. They also reduce the network traffic, since each popular object is requested from the original server only once (during lifetime of the object), after which the cache satisfies all future requests for the same object. Another advantage of web caching is the rapid deployment of content. As the web caches get widely-deployed, the content providers with popular data but without high-speed connections to the Internet, will be able to distribute their information to the users rapidly through these caches.

Web caching can be divided in three categories according to the location of the caches. The most common form of caching is done at the client side. The client-side caches store the most popular pages on the client's computer. Most of the modern browsers have this capability. The browser cache is useful for only the client and as long as the client continues to browse a web site which has multiple pages with

common objects. The main objective of the browser cache is to reduce the object retrieval latency.

Another type of caching used in the Internet is the server-side caching. In server-side caching an accelerator cache is located in front of one or more web servers. If the requested object is found in the cache, the accelerator returns the object, otherwise the request is routed to a back-end (origin) server. Server-based caching (or also called the reverse proxy) is implemented to avoid the overloading of the web servers. Reference [10] considers the problem of caching multiple web servers at a given location. The objective is to minimize the fraction of the requests forwarded to the back-end servers from the accelerator subject to fixed storage space. The paper models the problem as constrained-maximization problem and obtains the solution by Lagrange multiplier theorem.

Both client-side and server-side caching provide little help in alleviating the Internet congestion and server load. For this reason a third type of caching, the proxy caching is introduced. The proxy caches are placed between the clients and the original servers. They are implemented by the content providers, the enterprises or the ISPs. ISPs and enterprises implement proxies to reduce the WAN bandwidth without degrading the user performance. Content providers implement proxies to serve the user requests more rapidly and to avoid the overloading of the main server with large numbers of requests. Initial proxy servers that were implemented acted as the mirrors of the origin servers, where complete content was replicated.

The effectiveness of a proxy depends on its location. A web proxy located in a wrong place does little to improve the system performance. Finding the optimal placement of web proxies in a network like the Internet is a challenging if not an impossible task. Most existing proxies are placed in prominent locations such as the

routers for the LANs, ISPs' gateways, etc. The optimal location of web proxies depend on the user traffic and the network state. The determination of the best proxy locations in a network was investigated in [55], [56]. In [55] and [56], the authors explored the optimal placement of the multiple web proxies among potential sites, so that the overall user latency is minimized under a given traffic pattern. They showed that the optimal placement of proxies among potential sites in a linear or ring topology can be modeled as a dynamic programming problem.

The servers may install their own proxy caches or share the caches of the CDNs. Recently, the interest in CDNs has flourished due to the success of such CDNs as Akamai [93], MirrorImage [94], DigitalIsland [95], etc. These companies provide caching services for web servers. They are appropriately called *content providers*. Their promise is to improve the user performance significantly without having the content providers make huge investments in the mirror sites. Such a promise is made possible by many caching servers geographically dispersed all over the globe. These caching servers selectively cache documents from the host web content providers and use this content to serve user requests locally with less latency.

1.1.3 Cooperative and Hierarchical Caching

Multiple web caches placed at different locations in the Internet may cooperate and improve overall performance. For example, an institutional cache can be configured to send its HTTP requests to a cache in a backbone ISP at the national level. When the object is not available at the institutional cache, the request can be forwarded to the national cache. If the requested object is available in the national cache, it is served; otherwise the request is forwarded to the original server. The benefit of the hierarchical caching architectures is that higher-level caches have larger user population and

therefore higher hit rates.

The cooperation among the web cache sharing is first proposed in the context of the Harvest project [24]. An example of cooperative caching system is the NLANR [90] which consists of a number of backbone caches in the US providing service to institutional and regional caches. The caches obtain objects from each other using HTTP and ICP (Internet Caching Protocol). The Harvest group designed the ICP that supports discovery and retrieval of documents from neighboring caches. Today, ICP is widely used by institutions and countries that has established hierarchies of proxy caches which cooperate to reduce traffic to the Internet backbone.

Even if ICP is the de-facto standard for inter-cache communications, the large messaging overhead associated with the cache cooperation makes ICP unsuitable for wide-scale deployment for the web proxies. In ICP, a proxy cache discovers the content of other proxies by multicasting a query message to all other proxies whenever a cache miss occurs. Thus, as the number of proxies increases, both the communications and the processing costs increase dramatically. This query/reply exchange introduces additional delay as well. However, since the caches can process ICP queries very quickly, the delay is mainly due to the round-trip delay between the caches. The delay caused by the ICP depends on the caches' proximity to each other. Another deficiency of the ICP is its inability to deliver the objects to users from the optimal sites. An ICP proxy retrieves the object from the cache that has responded first to the inquiry. ICP does not consider the bandwidth available for the connection between the caches or the network states. Thus, it may be possible to retrieve the object from a cache, but that is located closer to the inquiring cache, which has low available bandwidth while there may be another cache which is located farther but has a high bandwidth link [22]. It is shown [73], [24], [26] that the use of ICP in a WAN creates large message

overhead and causes large additional retrieval delays. Often, users are better off by requesting the object directly from the original server.

Several alternatives have been proposed to address this problem, e.g. Cache Array Routing Protocol (CARP), Summary Cache, Cache Digest. CARP is specifically designed for clusters of caches implemented in a LAN environment, where multiple caches are placed mainly for scalability purposes. Often a single cache is not sufficient to handle the traffic or provide for the necessary storage space. Multiple caches share the load and provide a larger storage space. The problem of determining to which cache the user request should be forwarded to is then solved by CARP(Cache Array Routing Protocol) [91]. In CARP, URL space is partitioned by the help of hash functions. The proxy server or the client browser with CARP uses the destination URL of the user request in a hash function to determine the cache that the request should be forwarded to. When every user browser uses the same hash function, an object will never be present in more than one cache in the cluster. Unfortunately, this method is not appropriate for sharing caching resources in a WAN, since the caches have to coordinate the selection of the hash function. Such a coordination may be infeasible due to the limited network bandwidth among the proxies. Furthermore the network distance between the proxies and their users is usually non-homogeneous. However, CARP forwards the user requests only with respect to the destination URL, so some of the web objects may experience much higher retrieval delays.

Another proposal for this problem was given by two research groups simultaneously but independently: The Summary Cache [26] from University of Wisconsin and the Cache Digest [73] from NLANR. The proposed solution in both cases is to keep a summary of the contents of the neighboring proxy caches at each proxy. Thus, an incoming request is forwarded to a cache containing the requested object without

requiring a search for the object among the peered caches every time the object is requested. To keep the amount of information stored/transferred by the proxies low a technique called *Bloom filters* is suggested. The drawback of this technique is the possibility of false hits for the objects in the neighboring caches. That is, a user request can be forwarded to a neighboring cache even though that object is not stored there. However, numerical studies have shown that this technique can achieve comparable hit rates with reduced overall traffic and user latency compared to ICP [73].

Adaptive Web Caching [66] takes the idea of keeping the state of the neighboring caches one step further, and proposes URL routing. The idea is to forward the user request to a nearby cache which has high probability of having the requested object. However, in order to minimize the delay in the case of a *miss*, the request must be routed to a cache that is also closer to the origin server. Web caches maintain a URL routing table which is similar to IP routing table. The URL table's main key is the URL prefixes, with which one or more identifiers to the next-hop caches are associated. The information in the table includes a URL prefix, an identifier for the cache where the object is stored, and a metric reflecting that cache's average measured delay in seconds to retrieve a request from a matching URL. Such a system may be attractive for forwarding the requests of a few popular web sites, however this solution is not scalable. The amount of information that needs to be stored/transferred for large numbers of URLs seems prohibitive. Furthermore, the HTTP requests have to be constructed at the routers to determine the URLs of the pages, which may result in large processing delays.

LSAM project [78], and WebCanal [57] investigated the use of multicast channels to effectively deliver common information to distributed proxy caches. The LSAM (Large Scale Active Middleware) is a project on automatic distribution of Web pages

to proxy servers. Its proxy cache server tracks the requests and determines the common interests of the proxy caches. The proxy cache server forms multiple multicast channels according to these interests. Whenever a user request that belongs to one of these channels is received by the original server, the reply is sent to all proxy caches that are subscribed to that channel. By this way, the web pages are pushed to the clients anticipatively according to the requests of the peer proxy caches. When a user requests one of these pages, the page is already in the cache and can be served to the user immediately. Web Canal [57] generalizes this idea and suggests to use multicast channel not only for web page delivery but also for other push-based applications.

Cache Update Policies

Cache buffer sizes are limited and not all objects can be stored at every proxy. Thus, caches should try to store the most appropriate objects. The simplest and the most common approach to cache management is the LRU (Least Recently Used) algorithm, which removes the least recently accessed objects from the cache until there is sufficient space for the new object. The advantage of this algorithm is its simplicity. In fact, many current actual production cache engines and browser caches use some variation of this algorithm. However, one of the main weaknesses of LRU is that the cache can be flooded by objects that are referenced only once, flushing out objects with higher probability of being reused. The authors of [45] observed that the probability of an object being referenced again quickly grows after the second reference. Another widely implemented cache update policy is the Least Frequently Used (LFU). By disposing off the objects that are least frequently accessed, LFU avoids the shorthcoming of the LRU; however, LFU does not remove old objects with large reference counts from the cache, which result in *cache pollution*. An aging policy is often used to cope with this

problem.

The objective of cache management is to reduce latency and traffic. Due to the high variability and the complex structure of the Internet, achieving optimal solutions with respect to these objectives is almost impossible. For this reason, in the context of web caching several performance measures have been introduced. Three popular performance measures used in web caching are hit rate (HR), byte hit rate (BHR), and delay-savings ratio (DSR). These measures can be described as follows:

$$HR = \frac{\sum h_i}{\sum r_i}$$

$$BHR = \frac{\sum s_i h_i}{\sum s_i r_i}$$

$$DSR = \frac{\sum d_i h_i}{\sum d_i r_i}$$

where h_i is the number of hit references to document i, r_i is the total number of references to document i, s_i size of the document i, and d_i is the delay time to fetch document i from the original server to the cache. HR is the traditional measure for caching systems, and represents the number of hit references over the total number of references. BHR represents the number of bytes saved from retransmissions by using the cache over the total amount of bytes referenced. While BHR considers the size of the document, it does not consider the difference in retrieval costs. Among documents that are of the same size, those with higher retrieval costs should be kept in the cache. DSR is the measure that considers this issue.

The related research focused on determining an appropriate performance measure (e.g HR, BHR, etc) and optimizing the system with respect to this measure, so that the main objective of maximizing the throughput and minimizing the user latency

is also met. A good summary of web caching algorithms can be found in [4]. Several algorithms such as LRU, LFU, SIZE [83], HYBRID [84], LNC-R-W3 [75] have attempted to optimize performance for a particular measure. The weakness of each of these algorithms is their inability to adjust to new network and objective situations since they are designed for a single performance measure. More complicated algorithms such as Greedy-Dual-Size [15], LRV [60], sw-LFU [49], and LUV [4] have arbitrary objective functions which are not built into the cache replacement policy. The motivation for such algorithms is the need to provide different service levels to different content providers. Some servers will have clients who are much less tolerant to delay and are willing to pay for a higher quality of service. Some servers may be quite constrained in their network connections and thus may value off-loading traffic to a network cache. Basically what these algorithms do is that they assign a weighting function to each object in the cache. This weighting function has several parameters that can be adjusted to fit user needs. When needed, the algorithms replace the object that has the smallest weight. For example LUV algorithm uses a weight that is the retrieval cost of an object per unit size.

Recently Kelly et al. [50] discussed the user-centric design of web caching replacement algorithms. The authors explored a scenario in which the content providers reveal to a shared cache the value they receive from the hits to their objects. The authors proposed the Value Hit Rate (VHR) measure, which is a generalization of BHR measure. The servers associate each object i with a weight W_i to indicate the value they receive per byte when i is served from the cache. VHR metric is defined as

$$VHR = \frac{\sum_{hits} W_i \times \text{size}_i}{\sum_{requests} W_i \times \text{size}_i}$$

The drawback of such user-centered algorithms is that the servers may misreport

their valuations to receive larger share of the caching resources. Successful implementations of such designs require incentive mechanisms that induce truthful value reporting. These incentives can be provided by economic means (i.e., pricing of the cache resources). Reference [50] discusses the use of the generalized Vickrey auctions [81], [80] for truthful revealiation of the user valuations. In Vickrey auction, the winner of a single good auction pays the second highest bid. The bidder's announcement affects only when he wins, not how much he pays, and it can be shown that the bidder's dominant strategy is to bid his true valuation. Unfortunately, such auction-based strategies are not suitable for the large scale and highly dynamic structure of Internet.

Cache Consistency

For web caches to be useful, all caches should maintain an up-to-date copy of the objects. That is, all cached copies of the objects should be consistent. There are two types of consistency models: weak and strong [58]. In weak consistency a stale (non-up-to-date) document may be returned to the user, while in strong consistency no stale copy of the modified object is ever returned to the user after the server updates all caches. Weak consistency protocols include adaptive TTL (time-to-live) and client polling protocols. In the adaptive TTL approach a client considers a cached copy up-to-date if its time-to-live has not expired. The difficulty in TTL approach is in assigning TTL values to the objects. The adaptive TTL handles the problem by taking advantage of the the fact that a file is unlikely to be changed if it has not been modified for a long time. Thus, in adaptive TTL, the cache manager assigns a time-to-live attribute to a document, and the TTL is a percentage of the document's current age. Studies have shown that the adaptive TTL protocol keeps the probability of stale documents low ($< 5\%$) and it is argued that the adaptive TTL is the best

protocol for keeping weak consistency [37].

Strong consistency of the objects can be ensured by invalidation protocols. In this approach, the web server keeps track of all clients that store an object from the web server and when the object is updated, it sends invalidation messages to the clients. This approach may increase the server loads considerably, since the server needs to keep track of all clients with a copy of every object. However, hierarchical caching along with multicasting may increase efficiency of keeping strong consistency [72]. In [72] the authors compared pure push and automated pull strategies for varying degrees of staleness that is accepted by the users. It is shown that if the users can accept some staleness, automated pull with hierarchical caching is the optimal solution in terms of bandwidth usage and server load. However, if the users always require the most up-to-date documents, then the server gets overloaded by many pull requests. In that case, push scheme seems a better solution. The push scheme can be implemented with much less state information kept at the server and the proxies.

1.1.4 Implementation of Proxies

The users of a web site with multiple proxies need a mechanism to locate the nearby proxy caches. The basic method for end users to locate these proxies is to hand configure their browser with the proxies' URL, or with the URL of an *auto-configuration file*. However, this method is not scalable and is prone to errors. It would be much easier to manage, if the user browser automatically learns the configuration information for its web proxy settings. This problem is typically referred to as resource discovery problem [31].

The servers may use DNS redirection to inform the users, or they may have a URL page with a proxy auto-discovery script that a user may download to determine the

locations of the proxies. WPAD [31] is a protocol that is developed for the users to find this proxy auto-configuration file. WPAD protocol describes the methods that are used in a particular order to determine the auto-discovery URL. WPAD does not try to discover the name of the proxy server. The configuration file has other settings that allows other functionalities such as load balancing, request routing to an array of servers or automated fail-over to backup proxy server.

In order to completely avoid user reconfiguration, *transparent redirection* has been developed. Transparent redirection involves the interception and redirection of HTTP traffic to one or more web caches by a router or switch without requiring any coordination from the client. The non-HTTP traffic, and the traffic for the requests that cannot be satisfied locally are routed to the WAN.

WCCP [20] is a protocol that is developed for transparent caches to communicate among themselves. WCCP has two main functions: first, it allows a router enabled for transparent redirection to discover, verify and advertise connectivity to one or more caches; second, it also allows one of the caches (so called designated cache) to dictate how the router distributes redirected traffic across the cache farm.

Transparent proxy caching

A web cache is said to be transparent if the clients can access the cache without the need to configure their browsers, for example, without any need of either a proxy auto-configuration URL or a manual proxy setting. Transparent caches appear as a seamless part of the network architecture rather than a set of discrete proxy servers. The clients do not have to be aware of the existence of the proxies. Most ISPs prefer transparent caches, since these caches do not require an action from the clients.

A transparent caching system acts as a router and forwards to the WAN every-

thing but TCP traffic for HTTP requests (requests with TCP port number 80). A transparent cache accepts all TCP connections routed to it. When a client sends a TCP connection request for a web site to the router, the router forwards this request to the transparent cache rather than to the WAN. The transparent cache sends ACK to the client for this TCP request, and thus disguises itself as the original server.

However, this architecture is not scalable and is prone to system failures due to cache outages. To provide scalable and fail-safe operation, transparent caches are implemented together with *L4 (layer 4)* switches. These switches are called L4 switches because they base their switching decisions on the information in the TCP header, and TCP is a transport layer (layer 4) protocol in ISO OSI 7 layer reference model. L4 switches are connected to multiple caches for scalability. An incoming request's destination address in the TCP header is seeded into a hash function, which determines the cache that the request is going to be served by. The goal of a good L4 switch is to partition the URLs into non-overlapping clusters, so that each caching proxy serves certain number of distinct URLs. In the event of a cache failure, the switch forwards all requests that are destined to the failed cache to the WAN router.

DNS redirection

The clients need the IP addresses of the proxies in order to open a connection and retrieve the object. This information can be delivered to the clients during the initial DNS look-up. The original server may forward a list of proxy IP addresses to the client and the client may access one of these proxies in a round-robin fashion. However, with this method the clients are still unaware of the status of the proxies, and may find that the retrieval of the objects from their choice of proxy result in unsatisfactory performance.

More intelligent method of DNS re-direction is performed by the CDNs. The client looks up the IP address of the original server at the local DNS server. If the information is not available in the local DNS server, the request is forwarded to the authoritative name server of the original server. If the original server has subscribed to the replication service, it returns the CDN's central server's IP address as the authoritative DNS server to the local DNS server. The local DNS server then contacts the CDN's central server. The central server keeps statistics of performance of all of its proxies and their links. Upon receiving a DNS request, the central server checks the source IP address of the request to determine an optimal (in terms of physical distance, link congestion and load balancing) proxy cache that should serve the web request. The IP address of this proxy cache is returned as the DNS reply. Upon discovering the IP address of the proxy cache, the user requests the object from the proxy cache. The web page is downloaded from the local proxy server, if all parts of the web page is available at the local server. If not, the local server pulls the web page from the original web site on behalf of the user. Local server also decides dynamically if the pulled document should be cached or not according to the user traffic logs.

Recently, Rodriguez et al. [71] suggested the parallel use of all proxies for a single web page download to avoide the difficult problem of choosing the optimal proxy cache. The authors considered two parallel access schemes. In the *history-based* TCP access scheme, the client specifies a priori which part of the document should be retrieved from each proxy. This decision is made according to the observed rate of the servers that are calculated from the past accesses to the servers. In the *dynamic* TCP access, the client partitions the document into small blocks, and initially requests each block from different server. As a server finishes the delivery of a requested block, the client requests a new block from this server. The authors showed that such a parallel

retrieval results in near optimal delivery of the popular documents. However, this study assumed that the documents are large and they are identically replicated at all proxies.

DNS Overview

The principle task of DNS is to provide a mapping from the human readable domain names to the numerical IP-addresses used to identify the hosts in the Internet. DNS is implemented as a distributed database consisting of a hierarchy of name servers. The name space is divided into zones, where each zone has two or more authoritative name servers. A name server is authoritative for a host if it always has a DNS record that translates the host's hostname to that host's IP address. When a client needs to obtain an IP-address for a hostname, the client first sends the query to its local name server. Typically the local name server is close to the client, since each ISP usually implements a local name server. The local name server acts as the primary name server for the zone where the client resides, and has all the information about that zone as well as cached copies of the queries for hosts from other zones. Assuming that this name server does not have the requested information, it queries one of the root name servers. Currently there are 13 root name servers in the world that return the IP address of an authoritative name server that has the mapping for the requested host name. Then the local name server contacts the authoritative name server, and receives the IP address of the host, and finally this information is returned to the client's authoritative name server to be relayed to the client. [70], [54].

To avoid misdirection of DNS requests (that is, misdirection due to the client mistakes in entering the address or change of the IP address of the DNS server) web switches are used. The level 2 or 3 web switches determine which of the user messages

are DNS requests, and redirect these requests to a local or remote DNS server based on the ISP policy [92].

The initial DNS lookup may take several inquiry/reply transactions, and if the authoritative DNS servers are far from each other, this may lead to considerable delay for the users. Ref. [42] suggests to take advantage of recent advances in disk storage and multicast distribution to avoid this delay. In this approach, the geographically distributed and the so called *replicated servers* store the entire DNS database. To keep the replicated servers up-to-date, the new resource records are distributed by satellite broadcast or by terrestrial multicast. The replicated servers can be located at the local ISPs and at corporate and university networks. The DNS look-up procedure in this system is very similar to the original DNS look-up except that the local name servers send DNS inquiry to the closest replicated server instead of the root name server. The consistency of information among the replicated servers is preserved by the multicast delivery of new information as it becomes available.

1.1.5 Providing Content in the Internet

The publishers in the Internet (content providers) have several options in how they plan to reach the users. Publishers require a web site as a main source for information dissemination. Publishers' first option is to build their own web server. For this option, the content providers require a web server (a PC or workstation), an Internet router and a leased-line connection to an ISP. Furthermore, they have to pay for the Internet access to the ISP and they need to hire staff for maintaining the web server.

For scalability and security purposes, content providers may prefer to own their own web servers. However, instead of maintaining the servers at their location, they may co-locate them at an ISP's Network Operations Center (NOC). Consequently, content

providers still own dedicated servers for their web sites, but share the same (possibly high-speed) Internet connection with other content providers at the co-location host. These co-location centers are equipped with necessary security, backup equipment and staff to keep the servers in a good running condition. Publishers are charged for the amount of physical space their servers take up in the co-location site, as well as for the amount of user traffic served.

Many Internet publishers neither have the sufficient resources to build nor maintain their own web server. Consequently, *web content hosting* has become an increasingly common practice. Web content hosts own large amounts of resources (such as bandwidth, disks, processors, memory, etc.) and they offer to store and provide Web access to the documents from institutions, companies and individuals who lack resources. Publishers, who subscribe to this service do not actually own a web server, but their user traffic is directed to and served by the servers of the web content host. The storage and the bandwidth capacity of the servers of the web content host is shared among all web sites subscribing to the service. Usually, publishers vary in their expectations and requirements for the quality of the hosting service and the amount of money each is willing to pay. Thus, it is important to have the ability to offer different qualities of service to different customers. Most web servers today do not provide differentiated quality of service. Almeida et al. [1] investigated the priority-based request scheduling for providing differentiated QoS in web content hosts. The paper however, came short of investigating the optimal pricing schemes for different classes of users with varying requirements for resources (such as bandwidth, disk space, processing power, etc).

Content Distribution Networks

Due to the reasons discussed in the previous sections, ISPs and content providers may choose to disseminate the content to the caches throughout the network. While doing that they have several options: ISPs or content providers may install their own caches or out-source caching services to providers such as Akamai[93], Mirror Image[94] or DigitalIsland [95]. Not many content provider has the needs (due to large user traffic) or the resources to build or maintain its own global replication network. The high user traffic and alluring opportunities of global commerce over the Internet has thus lead to the development of the CDNs. CDNs have multiple caches with large storage and bandwidth capacities located in diverse geographical locations. For example, Mirror Image has built Content Access Points (CAP) where the caches are located. They also maintain a central server which continuously monitors the condition of the network links, and the proxy servers [94].

The vital component of a content distribution architecture is a method for redirecting clients to the proxy caches. The clients are usually redirected by a central server, which keeps (quasi-) real-time information on the conditions of the caches. There are two different redirection schemes currently employed by CDNs. Akamai [93] uses a selective redirection scheme, where the original server delivers the base HTML page to the client. The other objects in the HTML page such as large pictures are then delivered by the proxy caches. Mirror Image [94] uses complete redirection, where all objects including the base HTML page is delivered by an appropriate proxy cache. The advantage (and also the disadvantage) of the selective redirection is that the origin servers keep control of their web pages and dictate the objects to be replicated. They can also keep the history and statistics of user accesses, since the initial request is always served by the original servers. However, [43] argues that the selective redi-

rection is suboptimal compared to the complete redirection. The main reson for the suboptimality is that the user has to set up two different TCP connections: one to the original server for the base HTML, and the other to the closer proxy cache for the remaining objects. The second TCP connection to the proxy again suffers from the slow start and thus increasing the latency in the complete retrieval of the web page.

None of the CDNs favor hierarchical caches. This is due to the fact that the hierarchical caching not only may result in the use some other network's cache, but also may result in non-optimal routing and may cause multiple round-trip delays between the caches while searching for the cache containing the requested document. Thus, whenever a document cannot be found in the proxy server, the proxy server directly requests this page from the original server.

The E-business of Content Delivery

There is a strong demand from content providers for distributed network services that go beyond best-effort services. Current structure of Internet value-chain and the caching and replication technologies limit the content providers' control over their content especially with respect to performance and QoS. Current *value chain* is such that consumer picks up the content from the respective site through subscription to at least one ISP for reaching the Internet. In order to receive the content in a good quality, the user has to make QoS negotiations with the intermediary ISPs. In [59] a new *value-chain* is discussed, where the client subscribes to a content provider (e.g. e-newspaper) and to an ISP for a special service called *subscriber line management*. This service allows the content provider to control the QoS of the delivery of the content. Thus, the QoS negotiations are transparent to the user, because the content provider purchases communications services with appropriate QoS on behalf of the user. The

user only pays to the content provider and the content provider is responsible for the charges of the ISP.

The delivery of QoS guarantees in network bandwidth resources has been investigated extensively. However, especially for guaranteed web content delivery, the storage requirements should also be considered along with network bandwidth reservations. In [18] and [19] the *stor-serv* architecture is discussed for data storage services that can be considered as the dual of *intserv* and *diffserv* classes in the data transmission domain. Ref. [18] describes the stor-serv framework which consists of service specification and provision (resource reservation, resource mapping), resource management and discovery, security and economics. The content provider may specify the performance requirements in terms of access latency, jitter, acceptable miss rate, cost and bandwidth savings. These guarantees may be given either as deterministic or statistical. Obviously, statistical guarantees lead to better resource utilization [19]. The resource allocation problem is solved as *facilities-location problem* [19]. Combining the storage service allocation with network bandwith reservation for QoS may reduce the total cost of service provision for the content providers.

In fact Kelly and Reeves [52] considered a simple two-stage hierarchical caching model and solved the problem of determining the optimal cache sizes, when there is a tradeoff between the storage and bandwidth costs. However the authors did not consider the issue of QoS. In their model, the requests first arrive at the child caches. If the request cannot be satisfied there, then the request is forwarded to the parent cache. However, the bandwidth over the link connecting the parent cache and the child cache is not free. Thus each missed request incurs a cost for the content provider. Meanwhile, the storage space on the caches is not free either. The unit bandwidth and the cache prices are fixed and given as dollars per byte. Ref. [52] determines which documents

should be cached at which level of the hierarchy so that the overall cost to the content provider is minimized.

1.2 Current Pricing Structure in the Internet

Typical money flow in the Internet is from the end-users to the ISPs. The end-users pay their local ISP for their Internet service, who in turn pays for the interconnection to a regional ISP, who in turn pays for the interconnection to the national ISP. The national ISP has to pay costs of connecting the system to a data exchange location to communicate with other national ISPs. The cost structure of the services provided in any layer is determined by the prices charged by the providers one layer below and by the providers one layer above.

ISPs charge their customers according to two types of costs that they have incurred: access and usage costs. One type of access cost incurred by the ISPs is the installation cost. This cost refers to the set up of the ISP network including the investments in the network infrastructure. Another type of access cost is the customer activation costs, which refer to the costs associated with the connection of the customers to the ISP via modems, wiring, ISP's server disk space, IP address fee, etc [82].

ISP usage costs vary with customer. There are two types of usage costs: maintenance and network load costs. Currently, due to the flat interconnection fees the network load cost to ISPs is zero regardless of network congestion. However, the clients observe the network load cost as the delay when the network is congested. It is apparent that no two users have the same expectations from an Internet connection. Some users may download larger files, while some users may run applications requiring real-time information delivery. The users with real-time communications usually create more traffic and should be charged accordingly. Under a flat-fee scheme, low-usage

customers are subsidizing the heavy-usage customers by paying the same fee.

Eventually the ISPs expand their capacity to ease the congestion experienced by their customers. It is appropriate to expand capacity, if the marginal cost (cost of accomodating one more packet) is less than the marginal benefit (the price charged for the packet). The objective of the ISP is to cover this expansion cost through the customer usage fees.

1.2.1 Congestion Pricing

The congestion of the Internet causes wide range of dissatisfaction among the users. As discussed above, the network load has no cost for the ISPs, however; since the ISPs attract customers by the quality of their service, their market share is directly affected by their customers' satisfaction. Thus, the service providers may be inclined toward more sophisticated pricing schemes that can alleviate the network congestion.

Proper resource allocation plays a key role in improving the network performance. The centralized approach to resource allocation seems futile considering the large scale of the Internet, and the large variability of individual user's valuation of their connections. The pricing approach to resource allocation allows users to self-select the quantity that they are willing to purchase at the effective prices.

The current method of charging a flat-fee allows users to demand as much usage as they desire without any regard to the other users' connections. When everyone acts greedy, congestion may occur depending on the network capacity. Usage based pricing schemes are proposed to alleviate the network congestion. The social objective of a network is to maximize the total user satisfication. However, the current Internet community is very diverse, and many users will follow a non-cooperative self-maximizing behaviour. In that case, every user will try to maximize its own satisfaction by com-

peting with others for the limited resources. This problem can be described as a non-cooperative game, which has a *Pareto optimum* solution. In a pareto optimal solution, the exchange of the goods between the suppliers and the demanders is maximized by pricing the goods at their marginal cost. In other words, on the supply side prices should compensate suppliers (ISPs) of scarce Internet resources. On the demand side, prices should allocate resources efficiently by considering the value each user places on the Internet connection. However, also note that if there were no congestion, flat-fee pricing would optimally allocate current resources, since with the lines and routers already in place, there is negligible marginal private cost of operation [38].

The basic idea in pricing of network traffic is to charge the incremental cost of transporting the traffic over the network links. From the network's perspective, the incremental cost of transport of traffic through network is negligible. Thus, the incremental cost only depends on the *externality* cost. An externality is an effect of a participant on another that takes place outside the market [63]. In computer networks, the externalities include both congestion effects, where one user's use imposes a performance penalty on other users (congestion), and also connectivity effects, where a user benefits from other users being connected to the network (multicast). Current research has focused on externality or *congestion* pricing. Under this framework, a job is priced according to what impact it has on the QoS of other jobs.

An optimal congestion pricing scheme should depend on the current network load, the rate of transfer requested and the traffic volume of the session (amount of traffic generated during the session).

There have been several proposals for usage-based pricing schemes. Ref. [21] is one of the first papers that discusses the pricing issues in the computer networks. Ref. [21] discusses a simple two-priority service discipline and several different applications

running over the standard transport layer protocols. Thus, there is no specific QoS guarantee given to the users. The server basically keeps two FIFO queues for two service classes. Then, by simulation the authors compared the flat pricing with priority pricing, where a higher per-byte price is charged for the high priority traffic. The user satisfaction is measured in terms of the cost and the received QoS. The results suggest that every application type is better off with priority pricing, and the users when maximizing their utilities, also choose service levels that maximize the network efficiency. However, for the efficient use of resources, the charges for different priority classes should be updated as the load on the system changes. However, in the Internet the demand fluctuates frequently and it is not clear how the charges should be changed according to these fluctuations under this model.

Gupta et al. suggested a model that combines priority pricing schemes with the current fixed connection fee [35], [36]. The users specify a priority class when requesting a connection. The network services the requests of the higher priority classes first. A user pays a congestion toll and suffers a delay cost when accessing the network. The congestion toll depends on the marginal cost of the delay the user causes on other users. Each user tries to minimize the total of congestion toll and the expected delay cost. If the user valuation for the connection exceeds these costs, then it connects to the network, otherwise it waits for a less congested time to re-connect. This method distributes the load over time, and maximizes the social benefit. However, this proposal cannot also take into account the instantaneous fluctuations in the network state and thus cannot give optimal results for all users.

MacKie-Mason and Varian [64] proposed a "smart market" where each packet carries a *bid* in the packet header. The packets are given service at each router if their bids exceed some threshold, and each served packet is charged this threshold

price regardless of their bids. This threshold is chosen to be equal to the *market clearing* price. At market clearing price total demand equals to the supply. This is the point where the network resources are fully utilized. The main advantage of this proposal is that the users always bid their true valuation, since this only affects whether they get the service or not, but not how much they pay [80]. However, apart from the complexity, the proposal has the additional flaw of not achieving the true optimality. When the packet is denied of service at a route at a specific time, the packet is not lost. It is only delayed. This delay is unknown either to the user or to the network. However, users value their packets according to the delay they encounter, and thus for true bidding they need to know this delay in advance. Furthermore, the abovementioned procedure is repeated at every router, however; the true valuation for the user considers only end-to-end delay that a packet encounters. Extending the "smart market" framework for end-to-end connections is computationally prohibitive.

Ref. [65] describes another method of decentralized congestion pricing of a limited network resource with the objective of maximizing the efficiency of the network. The network efficiency is defined as the total user benefit less the cost of the network. The method internalizes the congestion by defining *shadow price* which is the total marginal congestion cost an increase in one of the user's share of the resource imposes on other users. However unlike Gupta et al., the authors assumed that the user utility functions are available to the network. MacKie-Mason et al. has also investigated the pricing schemes in a competitive environment. The authors assumed that the network implements two-part tariff for pricing: a fixed *subscription/attachment* fee, and a *usage* fee that depends on the total load generated by the user. They consider only linear pricing, where the network charges p dollars per byte user forwards. In their model both users and the providers try to optimize their benefit. Users choose

which provider to use and how much to use, while the provider chooses its capacity and how much bandwidth to supply to users. It is shown that a competitive supplier is forced to charge the socially optimal price for the quality of service that he offers. It is shown that at equilibrium the congestion in a system with no usage fee is higher than the one with usage pricing.

Pricing of the network resources (especially the bandwidth) has also been considered for improving the rate control algorithms used in the Internet (TCP) and the ATM networks. Kelly [47] introduced the notion of *proportional fairness*, which is basically equivalent to distributing the network resources to the users according to their valuations for these resources. The objective is to maximize the total social welfare of the users accessing the network with limited capacity. It is shown that this problem can be decentralized by individual user and network optimization problems that are solved independently. The coupling between the two optimization problems is the unit prices of the resources that the network advertises to their users. The unit prices of the resources are determined as the optimal solutions to the network revenue maximization problem under the current user demands. According to these prices users re-calculate the optimal demand that maximizes their net benefit. It is shown that there exists equilibrium resource prices and user demands that lead to the optimal solutions for the social welfare problem.

In [32] authors describe a method that can provide the end-users the necessary incentives to improve the network efficiency by marking appropriate packets at overloaded resources and by charging a fixed small amount for each mark received. This pricing method provides an alternative to the *Differentiated Services* architecture designed by IETF. Informing the end-users of the current state of the network by marked packets allows the end-users to design their own strategies in view of their personal

needs. Thus, the users do not have to choose among a number of fixed QoS classes. The user does not have to dynamically decide for their strategies as the network state changes. The decision of when and how much data to send can be made transparent to the user by implementing *intelligent agents*. Ref. [23] describes such an intelligent agent for Available Bit Rate (ABR) type connections in ATM networks. This work builds on the theory presented in [32]. Each user i declares his willingness to pay w_i, and is allocated a portion x_i of the available capacity. This portion is proportional to w_i ie $x_i = \frac{w_i}{\sum_j w_j} C$. Demand for bandwidth varies in time and the available capacity to ABR also changes due to VBR and CBR connections. Thus, offering the same amount of money does not lead to the optimal utility for money under different conditions. The intelligent agent (IA) determines the best willingness to pay function for varying conditions. The IA prior to the activation learns the user behavior by keeping record of the price that user pays for some bandwidth. Assuming that initially the user has the optimal amount of bandwidth with optimal willingness to pay (w_0, x_0), the network state has changed. In that case the user allocation for the same willingness to pay has changed to x_1. IA estimates the new price of the bandwidth by $p_1 = \frac{w_0}{x_1}$, and from the aforementioned price versus bandwidth function it determines the optimal willingness to pay. IA continually updates its willingness to pay, until an equilibrium is reached.

Instead of implementing intelligent agents at each user, the network itself can implement a *network broker* for the same purpose [29]. Fulp et al. proposes a network model at which the network links form individual competitive markets, where the link prices are updated according to the user demand by tatonnement process. In a tatonnement process the new price is equal to the previous price plus a correction function. The correction function increases the price when the total bandwidth used is beyond a threshold. If the total bandwidth in use is lower than this threshold the

price is decreased. The network broker calculates the amount of bandwidth that a user should request according to the effect that this bandwidth will have on the prices. If the user can afford the bandwidth under these new prices then the user starts transmitting at this rate. One simplifying assumption this study has made is that the portion of the user budget for each link in the route is given and fixed. However, dynamically proportioning of the user budget among the links should give better results.

1.2.2 Simplifying Congestion Pricing

Ref. [76] brings a new perspective to the optimal dynamic pricing schemes discussed so far. In this article, the authors argue that optimal congestion pricing may not be able to provide the necessary funds to cover the network costs, and in fact optimal congestion pricing may not be at all implemented due to its high complexity. For this reason, authors suggest approximate methods which may be suboptimal but can provide the sufficient network efficiency by simple to implement algorithms. First, the authors suggest to replace the cost of the actual path with the cost of the expected path, where the charge depends only on the source and destination of the flow and not on the particular route taken by the flow. This would help reduce the overwhelming communications among routers for each flow. Second, the authors suggest to approximate the current congestion conditions by the expected congestion conditions. This is essentially QoS-sensitive time-of-day pricing. Previous studies argued that time-of-day pricing has the problem that it does not reflect any instantenous fluctuations in traffic levels [67]. For example, packets that are sent during an idle network condition will be charged the full price as if there was congestion. However, as the authors argue users may overcome this problem by adjusting their service quality to the network condition. During low utilization, users may access the network via lower service class

but receive sufficient QoS and thus pay less.

Tsitsiklis and Paschalidis [79] discusses in detail the abovementioned issue of determining the right time scale over which prices should evolve. The authors investigated a problem, where there are multiple classes of users each of which demanding certain amount of network resources. The user requests arrive according to a stationary Poisson process. The users pay a fee depending on their class upon the acceptance of the request. The user arrival rates decrease as the fee increases. The objective of the network provider is to maximize either the revenue or the social welfare by determining a tariff as a function of available network capacity. It is shown that this problem can be solved as a dynamic programming problem. The authors also investigated the static (or time-of-day) pricing policies. The static pricing policies are of interest because they are simple to implement and provide users a predictable, fixed pricing structure. They have shown that when there are many users each with infinitesimal resource demand compared to the total capacity, the static pricing policy can achieve the optimum that the dynamic pricing policy achieves.

By approximating prices according to time and route, the resulting prices can be determined and charges assessed at the local access points. This local scheme is called *edge pricing* in [76]. Edge pricing schemes are very appealing, because they allow service providers offer many competitive and complex pricing schemes to the user without the overburden of complicated account keeping and messaging.

Chapter 2

Content Delivery Networks

The initial model for the World Wide Web was based on clients interacting with origin servers to request and receive content or services. As the Web increased in scale, this model proved unwiedly for several reasons and resulted in current industry efforts to build and operate CDNs. The purpose of these CDNs is to create a scalable service that can meet aggregate client demand while improving the performance and quality of delivery.

Content Networks typically aim to solve the "content distribution" problem, where the goal is to determine the best location between the original source and client for the content to be delivered. The best location of content usually depends on the network proximity and the load of the servers.

In the following, we first give a general formulation of the content distribution problem. Then, we describe the CDN architecture as will be considered in the rest of the thesis. According to this architecture, we develop a tractable version of the content distribution problem, and identify the important issues in this context.

2.1 Content Distribution Problem

Consider a network $N = G(V, E)$, where V is the set of vertices (nodes) and E is the set of edges (links) connecting the vertices. The vertices house the caching and processing resources, while the edges are the links between the network nodes, which have some fixed capacity bandwidth and buffer resources. There are M information servers that can be located at any of the vertices. Server m is the origin of information objects $\mathcal{M}_m = \{\mu_m^1, \mu_m^2, \ldots\}$. The objects of each server are replicated to other nodes in the network. Let $\mathcal{V} = \{O_1^{\mathcal{V}}, O_2^{\mathcal{V}}, \ldots, O_{|V|}^{\mathcal{V}}\}$ denote the sets of objects stored at each node (a replication strategy). Let $O_v^{\mathcal{V}} = \{o_1^v, o_2^v, \ldots\}$ denote the set of objects stored in the node v under the replication strategy $\mathcal{V}$. Let $s_{o_k^v}$ be the size of the object o_k^v stored at node v. Then $\sum_{k \in O_v^{\mathcal{V}}} s_k \leq C_v$, where C_v is the size of the cache at node v.

There are K users requesting objects from the servers. Assume that user i requests object j_i and it is delivered from node v_i. Let $\mathcal{P}^{i,v_i}$ be the set of paths between the node user i resides and the node v_i. A path $p_i \in \mathcal{P}^{i,v_i}$ is selected for the delivery of the object. The path p_i defines an ordered set of links l from node where user i resides to the node v_i, i.e. $p_i = \{l_{p_i}^1, l_{p_i}^2, \ldots, l_{p_i}^{L(p_i)}\}$, where $L(p_i)$ is the length of path p_i. Let r_i^l be the bandwidth resource allocated to user i's request over the link $l \in p_i$. Define $R_i(p_i) = \{r_i^l, l \in p_i\}$. A resource allocation is feasible, if at no link the total resources required by the connection is more than those available. Assume that total bandwith resource available on link l is B_l.

User i receives a utility $u_i(j_i, v_i, p_i, R_i(p_i))$ from object j_i, when it receives the object from node v_i over the path p_i where $R_i(p_i)$ bandwidth resources are allocated over the links of the path. The social objective of any network is to maximize the total user utility. The content distribution problem can be described as a joint object

replication and resource allocation problem. Let $\eta(l, \mathbf{p}) = \{i : l \in p_i, i = 1, 2, \ldots K\}$ be the set of users with their paths containing link l. Then the joint optimization problem is given as follows:

$$\max_{\mathcal{V}} \max_{\{p_i \in \mathcal{P}^{i,v_i}, R_i(p_i)\}} \sum_{i=1}^{K} u_i(j_i, v_i, p_i, R_i(p_i)) \tag{2.1}$$

subject to

$$\sum_{i \in \eta(l,\mathbf{p})} r_i^l \leq B_l, \qquad \forall l \in E$$

$$\sum_{o \in O_v^{\mathcal{V}}} s_o \leq C_v, \qquad \forall v \in V$$

This is a combinatorial optimization problem with many state variables. It has been shown in the literature that simpler versions of this problem are NP-hard. We propose to develop distributed methods for finding a near-optimal solution to this problem. We resort to microeconomic methods for finding the distributed solutions. Two basic microeconomic approaches have been discussed in the literature for finding efficient distributed resource allocation schemes in the networks: resource-directed and price-directed [40]. In the resource-directed approach, each user calculates the marginal values for its current resources and communicates it to the other users. The allocation is then changed, so that the users with above average marginal utility receive larger portion of the resource. On the other hand, the price-directed approach sets an initial set of prices for the resources, which is then announced to the users. The users determine their resource allocation requests according to these prices. Prices are then iteratively changed to accomodate the demands for the resources, until the total demand equals to the total resource available.

In this work we are going to use the price-directed approach, since it more closely

models the real systems.

2.2 Content Delivery Network Architecture

We can conceptualize content networks as the interconnection of network elements at the application layer of the OSI model. Whereas lower-layer network infrastructures revolves around the routing, forwarding and switching of frames and packets, content networks deal with the routing and forwarding of requests and responses for content. The units of transported data in content networks are often very large and span hundreds or thousands of packets.

Currently there is a considerable effort in developing an architecture to peer content networks to improve the Internet performance [25], [34], [68], [2], [6] and [14]. In this effort, the trend is moving from the speculative caching (proxy servers) to the service level agreements (SLAs) between the publishers and the surrogates. The *publisher* is the party that ultimately controls the content and its distribution. A *surrogate* is a delivery server, other than the *origin server*. An *origin server* is the point at which the content first enters the Internet. The origin server for any object is the server or a set of servers that holds the authoritative copy for that object. The user requests are routed to the surrogates, which deliver the corresponding content. Publishers have more control over their content by using the surrogates. The desired level of quality of service can be defined with respect to variety of different parameters such as average delay experienced by the users and the web site's server load or the amount of space allocated to a web site on the surrogates and the extend of geographical distribution of content, etc.

In the on-going work, [2], [34], [68] with respect to the peering of the CDNs, a simple architecture has been developed. In this architecture a content network is

modeled as composed of three seperate sub-systems:

1. *Distribution System* coordinates the activity of moving publisher's (content provider's) content to one or more of the surrogates. Content dissemination can be publisher-initiated ("push" of the documents according to anticipated user load) or CDN-initiated ("pull" of the documents after receiving a client request) or both.

2. *Request-Routing System (RRS)* coordinates the activity of directing a client request to a suitable surrogate. RRS may direct the request to one of the surrogates of the CDN, to a peered CDN or the original server itself. The selection of the most suitable server depends on the load, availability and user preference-location, delay, etc. of the surrogate cache. As discussed in detail in the later sections, we argue that these factors can be incorporated in a shadow price announced by the surrogate.

3. *Accounting System* determines the methods for measurement and pricing of the distribution and delivery activities. Usually the CDN may charge for two resources: storage and bandwidth. Storage charges may correspond to the activity of distribution the publishers' content. Meanwhile the bandwith cost relates to the user requests that are serviced by the CDN.

Figure 2.1 illustrates the CDN architecture that we will consider in this thesis. Operation of the system can be summarized as follows:

1. Publisher selects a desired QoS and negotiates a Service Level Agreement (SLA) with the CDN. The selected level of QoS depends on the benefit received by the publisher/clients and the cost of service with that level of QoS. The dissemination of the content to the surrogates takes place according to the anticipated user loads. The user load distribution can be estimated according to a prioiri request

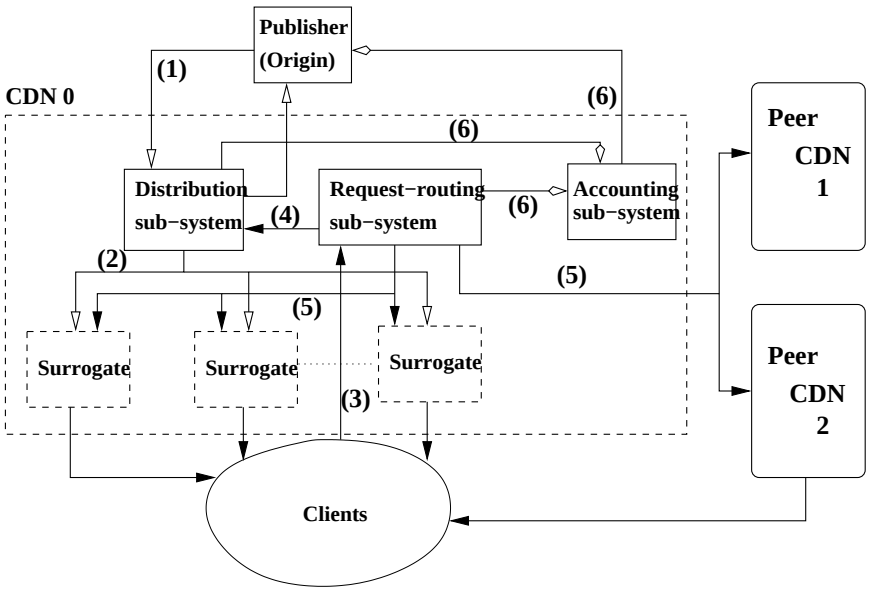

Figure 2.1: CDN Architecture

statics as well as other off-line market research. Either publisher or the CDN determines which content has to be disseminated to where according to the user statistics, the desired QoS (in terms of average user latency, server load, etc.), the locations of the surrogates and the CDNs' pricing methods.

2. Distribution sub-system coordinates the dissemination of the content to the surrogates in lieu of a SLA, which is agreed upon by the publisher and CDN. The distribution decision requires the information about the publisher utility functions, user access statistics and the geographical locations and storage and transmission resource capacities of the surrogates.

3. Client requests are directed to a unique network element responsible of request routing. The RRS can be implemented in several forms:

   (a) DNS re-routing: The client is informed about a suitable surrogate by DNS resolution. Client IP address and full URI are not inspected to determine

a suitable surrogate.

(b) Application-level routing: full URI as well as the client IP address are inspected to determine the most suitable surrogate at the cost of increased processing delay.

(c) The request routing function may be implied by an in-path network element such as caching proxy, which is typical for a Access Content Network. In this case, the request routing is optimized to a null function, since the client is a priori mapped to a surrogate. If the object is locally stored, the request is immediately served, otherwise the request is forwarded toward the original server. There may be several in-path elements (hierarchical caching) performing the same operations.

An important issue for DNS and application-level routing is their substantial information needs for correct functioning. These methods require the up-to-date content and availability (network and server) knowledge from all the surrogates as well as the peered CDNs.

4. Distribution sub-system may re-examine the content of the surrogates according to the current request arrival distribution. Either surrogates may request the objects upon cache-miss similar to the way current proxy caches do, or these statistics can be used by the publishers or the distribution system (as an agent for publisher) to periodically re-arrange the content to satisfy the desired SLAs.

5. Request-routing system may forward the client request to a peer CDN in case the requested object cannot be found in its own surrogates and/or it is more efficient to do so. Peered CDNs appear as black-boxes to each other and can only gather information about their distribution and request routing sub-systems

according to the content and network advertisement that they send out periodically. In these advertisements CDNs announce the network state (including geographical locations, capacity, load and the content stored in their surrogates) and the summary of the available content of each surrogate. Real-time update of these advertisements can be difficult due to the difficulty to dynamically measure network-state and the size of the content information in the surrogates. However several methods can provide approximate behavior, e.g. bloom-filters [26], delta updates [73], periodical polling [28], etc.

6. The records for caching and transmission costs are kept for accounting purposes. CDN charges the publishers for the amount of content stored in their network. This charge may depend on the SLA. The storage charge may refer to the shadow price for the limited storage space available in the surrogates. Furthermore, the CDN may charge for the individual requests served by the network. This charge can be validated by the limited processing capacity of the surrogates.

2.3 Content Delivery Problem

We have seen that the *content distribution problem* is a combinatorial optimization problem, which is very difficult to solve optimally. In this section, we consider a simplified version of this problem that also fits more closely to the current content network architecture.

We consider a business model in which the primary customers of the CDNs are the web servers (publishers). Notice that the benefit of each publisher can be given as the total utility of the individual users accessing the publisher. According to the notation

used in previous section, the utility of publisher m, U_m, is given as

$$U_m(\mathbf{v}, \mathbf{p}, \mathbf{R}(\mathbf{p})) = \sum_i \sum_{j_i \in \mathcal{M}_m} u_i(j_i, v_i, p_i, R_i(p_i)), \tag{2.2}$$

where $\mathbf{v}= \{v_i\}_{i=1}^K$, $\mathbf{p}= \{p_i\}_{i=1}^K$ and $\mathbf{R}(\mathbf{p})= \{R_i(p_i)\}_{i=1}^K$ are the vectors of cache locations, paths and resources allocated on these paths used for the delivery of the users' requested objects respectively.

The *system* objective is to maximize the total benefit of the publishers subject to the limited caching and bandwidth resources. The joint system optimization problem in eq. (2.1) can be re-written in terms of publisher utilities as

$$\max_{\mathcal{V}} \max_{\{p_i \in \mathcal{P}^{i,v_i}, R_i(p_i)\}} \sum_m U_m(\mathbf{v}, \mathbf{p}, \mathbf{R}(\mathbf{p})) \tag{2.3}$$

subject to

$$\sum_{i \in \eta(l, \mathbf{p})} r_i^l \leq B_l, \qquad \forall l \in E$$

$$\sum_{o \in O_v^{\mathcal{V}}} s_o \leq C_v, \qquad \forall v \in V.$$

In this work, we consider the optimization of abovementioned *system* optimization problem as well as the optimization of the multi-criteria problem formed by the maximization of the individual publisher net benefits. Notice that in the Internet agents (users, publisher, caches, etc) are selfish and non-cooperative and try to maximize their own benefit regardless of other publishers. We will investigate the effects of such behaviors as compared to the *system* solution.

We consider the average performance of the system. The system objective is to maximize average aggregate publisher utilities. We assume that URL routing is used to direct user requests to the appropriate surrogates. However, only the original server

identifier is used in the routing decision and not the complete object identifier. We believe this type of operation is more efficient due to the high processing and messaging costs affiliated with the complete HTTP header routing. Notice that for complete HTTP header routing, the complete and up-to-date cache content information is required. Although schemes such as delta update and summary cache [26] are suggested, they are not very practical and easily implementable. Thus, when a user request is directed to a surrogate, there is a probability that the requested object is not found in the cache. In that case, the request is re-routed to the original server. We do not consider hierarchical/cooperative caching in this architecture, since hierarchical/cooperative caches result in increased processing and messaging overhead.

Users are clustered in LANs. They connect to the publisher web sites via their ISPs and the WANs. Assume that a user from network n is interested in an object provided by the publisher m. Also assume that the user request is routed to the surrogate s. The probability of the user request served at this surrogate depends on the content dissemination strategy of the publisher m, which can be in effect summarized by the cache hit probability Pr^m_{hit}. The total retrieval delay is the sum of the propagation and transmission delays. The average propagation delay that the user expects is $d^{s,prop}_{n,m} = d^{prop}_{n,s} + (1 - Pr^m_{hit})d^{prop}_{s,m}$, where $d^{prop}_{k,l}$ denotes the propagation delay between the nodes k and l. The average transmission delay depends on the current load of the servers and is given by $d^{s,tran}_{n,m} = Pr^m_{hit}d^{tran}_s + (1 - Pr^m_{hit})d^{tran}_m$, where the transmission delay corresponding to the current load at node k is assumed to be known as d^{tran}_k. We assume that all objects are the same size. The network distance, i.e. the delay between two nodes, can easily be determined by methods such as periodic probing [28]. The total average delay expected by the user is,

$$d^{s,total}_{n,m} = d^{s,prop}_{n,m} + d^{s,tran}_{n,m}. \tag{2.4}$$

A reasonable user utility function decreases with increasing latency $d_{n,m}^{s,total}$. Thus, the primary objective of the content delivery problem can be summarized as the minimization of a function of the average user latency.

For our purposes we only consider the bandwidth and caching resources in the CDN. We assume for fairness reasons that every user request is assigned the same amount of bandwidth. Thus, each cache can serve a fixed number of user requests at a time. The requested objects are delivered to the users from the selected cache over the shortest path in order to maximize the user utilities.

According to the definition of content delivery problem as given in eq. (2.3), we may divide the content delivery problem into two: distribution and request routing sub-problems. The distribution sub-problem solves the optimization problem (2.3) for the optimal distribution of objects given the rate of user requests arriving to each cache. The routing sub-problem solves (2.3) for optimal user request arrival rates to each cache given the object dissemination strategy. Although these two sub-problems are jointly related, for the purpose of designing a practical algorithm we envision an iterative scheme in which each sub-problem is solved separately. In this scheme, the distribution sub-problem determines the dissemination strategy according to the arrival rates as determined by the request-routing sub-problem. The request routing sub-problem determines the routing strategy according to cache content as determined by the distribution sub-problem. These sub-problems update their decisions iteratively according to the output of the other. We expect that if the solutions to these two sub-problems are (near-) optimal, then the solution to the overall content delivery problem is also (near-) optimal.

In order to allocate limited resources as efficiently as possible, we implement price-directed market-based algorithms. Thus, each surrogate charges a price for the amount

of resources allocated to the publishers. In the following, we define the distribution and routing sub-problems with the understanding that each publisher and surrogate maximizes its own benefit regardless of the system optimization problem. Notice that the benefit of a publisher is the total utility of its users with the given resource allocations less the cost of these resource allocations. Meanwhile, the benefit of a surrogate is the total revenue received by selling/renting the available resources. In the later sections of the thesis, we show that the competitive behavior leads to solutions that are close to the system optimum solution.

2.3.1 Distribution sub-problem

Let $\lambda^n_{m,s}$ be the request arrival rate observed at surrogate s for the objects in the publisher m that are received from the users in LAN n. Let $\lambda^n_m = \sum_s \lambda^n_{m,s}$ be the total request arrival rate for the objects in publisher m from the users in LAN n. Also let $\lambda_{m,s} = \sum_n \lambda^n_{m,s}$, be the arrival rate to surrogate s for the objects in the publisher m. For the purposes of the distribution sub-problem, we assume that $\lambda^n_{m,s}$ is known. The objective is to disseminate the publisher's content to the surrogates so that the net publisher utility is maximized. In the previous section, we mentioned that the content dissemination strategy of the publisher can be summarized by the cache hit probability. Assume that $x_{m,s}$ amount of caching space is allocated for the objects of the publisher m in the surrogate s. In order to maximize the cache hit probability the publisher disseminates the most popular objects to the surrogate. Let $Pr^m_{hit}(x_{m,s})$ denote the cache hit probability observed when $x_{m,s}$ caching space allocated to the publisher.

Further assume that users have an identical utility function, $u(v, p_{n,v}, r)$, where v is the surrogate delivering the object, $p_{n,v}$ is the shortest path between the surrogate

and the user network and r is the fixed resource allocated for any user request. Thus, the utility function for publisher m depends only on the sizes of the cache allocations in each surrogate, $x_{m,s}$ and the user request arrival rates to each surrogate, $\lambda^n_{m,s}$, i.e.

$$U^s_m(x_{m,s}, \lambda_{m,s}) = \sum_n \left[\lambda^n_{m,s} Pr^m_{hit} u(s, p_{n,s}, r) + \lambda^n_{m,s}(1 - Pr^m_{hit}) u(m, p_{m,s}, r)\right]. \quad (2.5)$$

In the distribution sub-problem, the only cost arises from the cost of usage of caching resources. Let p^{cache}_s denote the price charged by surrogate s for the unit cache space. We assume that there is no collaboration among the publishers or publishers. Each publisher and surrogate tries to maximize its own net benefit regardless of others. The optimization problem solved by the publisher m can be given as:

$$\text{(Distribution/Publishers)} \ \max_{x_{m,s}} \sum_s U^s_m(x_{m,s}, \lambda_{m,s}) - \sum_s p^{cache}_s x_{m,s}, \quad (2.6)$$

Meanwhile, surrogate s solves the following optimization problem:

$$\text{(Distribution/Surrogates)} \qquad \max_{p^{cache}_s} \sum_m p^{cache}_s x_{m,s} \quad (2.7)$$

$$\text{s.t.} \qquad \sum_m x_{m,s} \leq C^{cache}_s,$$

where C^{cache}_s is the total caching capacity of the surrogate.

This type of system is called a game. The publishers and surrogates are playing a game in which each try to maximize its benefit by selecting an appropriate strategy. However, the final benefit received depends on the strategies of other publishers and surrogates. We analyze this system in Chapter 3. We determine the optimal strategies of the publishers and the surrogates. We also show that this game has an equilibrium strategy, where no player can change its strategy without reducing its benefit.

2.3.2 Request-Routing Sub-problem

The objective of the request-routing sub-problem is to direct a user request to an appropriate surrogate. We may conceptualize the RRS operation as depicted in Figure 2.2. The requests from the users are first intercepted by the RRS. The RRS is basically a DNS server performing URL routing. The RRS checks the HTTP header to determine the user network and the requested URL. The RRS also keeps real-time information on the network distances between the user networks and the surrogates. The RRS has the knowledge of which surrogates contain (at least with high probability) the objects requested by the clients. Thus, for each object requested by the clients we may have different set of surrogates $S^0, S^1, \ldots, S^K$. The RRS may route the user request directly to the origin server, S^0, if it is optimal to do so. Surrogates charge p_s^{tran} for each user request served. This price can be interpreted as the shadow price for the load on the surrogates. Publisher i optimization problem can be given as:

$$\text{(Routing/Publishers)} \ \max_{\lambda_{m,s}} \sum_s U_m^s(x_{m,s}, \lambda_{m,s}) - \sum_s p_s^{tran}\, \lambda_{m,s}, \tag{2.8}$$

where $\sum_s p_s\, \lambda_{m,s}$ is the cost of serving $\lambda_{m,s}$ user requests at the surrogate s.

Meanwhile, surrogate s solves the following optimization problem:

$$\begin{aligned} \text{(Routing/Surrogates)} \qquad & \max_{p_s^{tran}} \sum_m p_s^{tran} \lambda_{m,s} \\ \text{s.t.} \qquad & \sum_m \lambda_{m,s} \leq C_s^{tran}, \end{aligned} \tag{2.9}$$

where C_s^{cache} is the total caching capacity of the surrogate.

In Chapter 4, we determine the solution to the abovementioned game. The routing game formed by above two optimization problems is shown to have an equilibrium as well, from similar results given in Chapter 3. Since these two sub-problems have

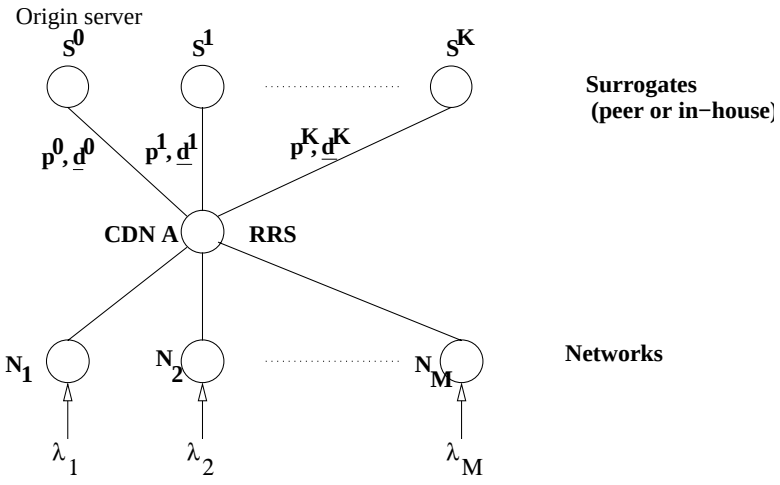

Figure 2.2: Request-routing sub-system

an equilibrium, and the equilibrium solutions are continuous, we establish that there exists an equilibrium to the overall problem.

In Chapter 4, we also consider a different objective for the RRS. In many cases, instead of requesting separate resource allocations from each surrogate, the publishers may request from the CDN an average delay bound for their users' requests to be satisfied. In such a case, the objective of the RRS is to satisfy this delay bound at the minimum total cost to the publishers. Such an objective makes sense in the competitive business model of the Internet. In this model, the CDN optimization problem is given as:

$$\text{(RRS)} \qquad \min_{\lambda^n_{m,s}} \sum_m \sum_s \sum_n p^{tran}_s \lambda^n_{m,s} \tag{2.10}$$

$$\text{s.t.} \qquad \sum_s \sum_n \lambda^n_{m,s} d^{s,total}_{n,m}(x_{m,s}) \leq \mathcal{D}_m, \; \forall m,$$

$$\sum_m \sum_n \lambda^n_{m,s} \leq C^{tran}_s, \; \forall s,$$

where C^{tran}_s is the total transmission capacity of the surrogate. In Chapter 4, we

determine the solution to this problem. We also show that the optimization problem in (RRS) usually does not have an equilibrium, when the publishers update their requested delay bounds according to the CDN prices.

2.3.3 Provision of *diff-serv*-like QoS in CDNs by Nonlinear Pricing

The distribution and request-routing sub-problems together establish a system, where the publishers can adjust their QoS with infinite precision. We also explore the case, where the CDN provides only limited number of options for QoS. In Chapter 5, we explore the optimal pricing rules when the available resource is shared among the users. The resource provider partitions its resource and charges a different price in each partition. The users subscribe to a partition, and share the resource available in the partition with those who also subscribed to the same partition. The user's decision to subscribe to a partition depends on the net benefit it receives. A similar type of system was investigated by Odlyzko in [69]. Unlike [69], we investigate the optimal nonlinear pricing scheme, when the user utility functions can be estimated. We further investigate the optimal condition for partitioning the resources. We determine that the optimality of resource partition depends on the user arrival rate, the capacity of the resource and the net benefit received by subscribing to the resource.

Chapter 3

Content Dissemination

In this chapter, we consider a realistic model for the relationship between the publishers and the content delivery networks. The publishers disseminate a part of their content to the surrogates to improve the user latency. Meanwhile, surrogates charge the publishers for the amount of caching space the publisher's content allocate. However, there are multiple surrogates competing to serve the publishers. We investigate the effect of this competition on the system. Specifically, we show that such a price competition leads to an equilibrium, which under certain conditions, leads to the optimal cache allocation strategy for the publishers. This approach provides a dynamical and distributed algorithm for determining the cache content in the network, which has a performance close to the optimum solution.

3.1 System Model

Figure 3.1 illustrates the network set-up that we are interested in this chapter. There are several LANs where the users reside. Every user is interested in one or more of the objects of a publisher. If the publishers have subscribed to a CDN, the user requests are first intercepted by the request-routing sub-system of the CDN. The request-routing

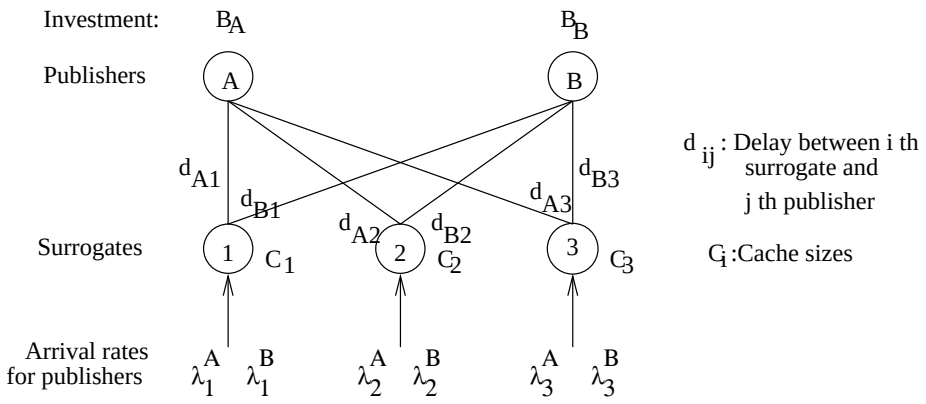

Figure 3.1: Content delivery system

subsystem then forwards the user request to an appropriate surrogate, perhaps the one with the highest probability of delivering the requested object with minimum latency. For the distribution sub-problem, we assume the routing decision is made and the user request arrival rates to each surrogate is fixed. The surrogates are located between the user networks and the origin servers. Thus, users are always at most two hops away from the content. A user request is first checked at the corresponding surrogate. If the requested object is available at the surrogate, the request is immediately served. Otherwise, the request is forwarded to the origin server, where the object originally resides.

The surrogates have limited cache size, and the cache is shared among the publishers. The surrogates charge the publishers for the portion of the cache their content occupies. We assume that the surrogates of a CDN does not cooperate, probably due to high messaging and processing overheads associated with cooperative caching. Instead, they act non-cooperatively with the objective of maximizing their individual revenues. The surrogates compete among each other to store the publishers content. The publishers do not collaborate either, and try to purchase as much cache space as possible with the objective of maximizing their net benefit.

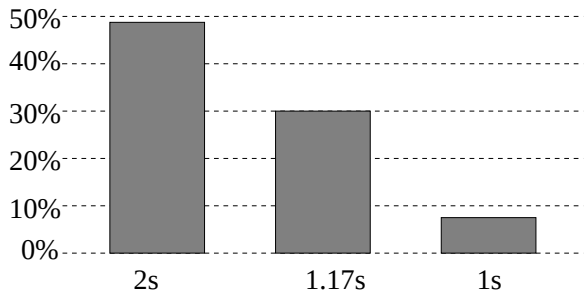

Figure 3.2: Bail-out rates for various download times [89]

Assume that there are I different publishers (origin servers) and J different surrogates present in the network. The user requests arrive from N different user LANs. Let $\lambda_i^{j,n}$ denote the total request arrival rate from LAN n at surrogate j for the content in the ith publisher. Let $\lambda_i^j = \sum_n \lambda_i^{j,n}$ be the total arrival rate to the surrogate j for the content in publisher i. The user interest in the objects of the publishers is distributed according to Zipf distribution [88]. That is, given that a request has arrived, the probability that the request is for object h is $q(h) = \frac{c}{h^{\alpha_i}}$, where c is the normalization constant, and $0 < \alpha_i < 1$ is the distribution characteristic of publisher i. The characterization of user request distribution as a Zipf distribution is discussed in previous studies [13], [51] and is widely accepted as a good approximation to the actual web traffic behavior.

Recent studies [89] have shown that as the latency increases, users stop browsing the requested page (*bail-out*) with increasing probability (Figure 3.2). Since the Internet became more and more commercially oriented, the bail-out rate started to have a direct economic impact for the web sites. It is of interest for web sites to have a fast object delivery rate in order not to lose customers. The web sites (publishers) can be considered as content providers. They make their revenue either by selling some information content (such as news, maps, etc.) or by selling tangible products. The

necessary (but not sufficient) condition to keep the users browsing the web site is the timely delivery of the content. However, one can easily see from Figure 3.2 that the user bail-out rate have nonlinear relationship with the retrieval time. Thus, from the content providers' view the important metric is the minimization of the lost revenue rather than the retrieval time. In this chapter, we consider a more generalized version of the content delivery problem, where every publisher i receives varying benefit from provision of a user request with a certain delay, which is reflected in the selection of benefit function $w_i(d)$. We assume that $w_i(d)$ is concave.

We can identify two types of optimization problems in this model: publisher's revenue maximization and the surrogates revenue maximization. We first determine the publisher's optimal caching strategy under a certain surrogate pricing scheme.

3.2 Optimal Publisher Strategy

Let B_i^j be the investment of the ith publisher in the jth surrogate. Let $\mathcal{B}_i = \sum_j B_i^j$ be the total investment of the ith publisher. It is assumed that the information stored in the publishers is continuous and can be replicated continuously to a surrogate. The total information available at the publisher i is χ_i. The publisher replicates its most popular part of the content to the surrogates so that the cache hit probability is maximized. Assuming that C_i units of cache space is allocated to the publisher, the probability that an incoming user request is satisfied at the surrogate is given by,

$$\begin{aligned} Pr(\text{hit}|C_i) &= \int_o^{C_i} q(x)\,\mathrm{d}x = \int_0^{C_i} \frac{1-\alpha_i}{\chi_i^{1-\alpha_i} x^{\alpha_i}}\,\mathrm{d}x, \\ &= \left(\frac{C_i}{\chi_i}\right)^{1-\alpha_i}. \end{aligned}$$

Note that we assumed that $q(0) = 0$ in arriving this result.

Let p_j denote the price of the unit cache space in surrogate j. Let the pricing policy, $\mathbf{p} = (p_1, p_2, \ldots, p_J)$, denote the set of unit cache space prices of all the surrogates in the network. Let d_{ij} denote the additional average delay a user request forwarded from surrogate j to the origin server of publisher i will experience. Let x_i^j be the cache space allocated to publisher i in surrogate j. If i th publisher's investment in the j th surrogate is B_i^j, then the total cache space allocated to the content of publisher i in surrogate j is $x_i^j = \frac{B_i^j}{p_j}$. The average reduction in the user delay or equivalently the average net benefit that publisher i generates by B_i^j investment in surrogate j is $\lambda_i^j w_i(d_{ij}) \left(\frac{B_i^j}{p_j \chi_i}\right)^{1-\alpha_i}$. Define $\beta_i^j = \lambda_i^j w_i(d_{ij})/\chi_i^{1-\alpha_i}$ as the gain factor for publisher i from surrogate j.

The utility function, i.e., the total *additional* average benefit, $U_i(x_i)$, of publisher i is $U_i(x_i) = \sum_{j=1}^{J} \beta_i^j \left(x_i^j\right)^{1-\alpha_i}$. For a given pricing policy $\mathbf{p}$ the publisher optimization problem (S) can be written as:

$$\text{(S)} \quad \max_{\{x_i^j\}_{j=1}^{J}} \quad U_i(x_i) \tag{3.1}$$

$$\text{subject to} \quad \sum_{j=1}^{J} x_i^j p_j \leq \mathcal{B}_i.$$

Since $U_i(x_i)$ is a concave function and the constraint set is compact, there exists a unique solution to (S).

Lemma 1 $x_i^{j*} = \dfrac{\left(\frac{\beta_i^j}{p_j}\right)^{1/\alpha_i} \mathcal{B}_i}{\sum_{k=1}^{J} p_k \left(\frac{\beta_i^k}{p_k}\right)^{1/\alpha_i}}$ *is the unique optimal solution to the optimization problem (S).*

Proof See appendix A. ■

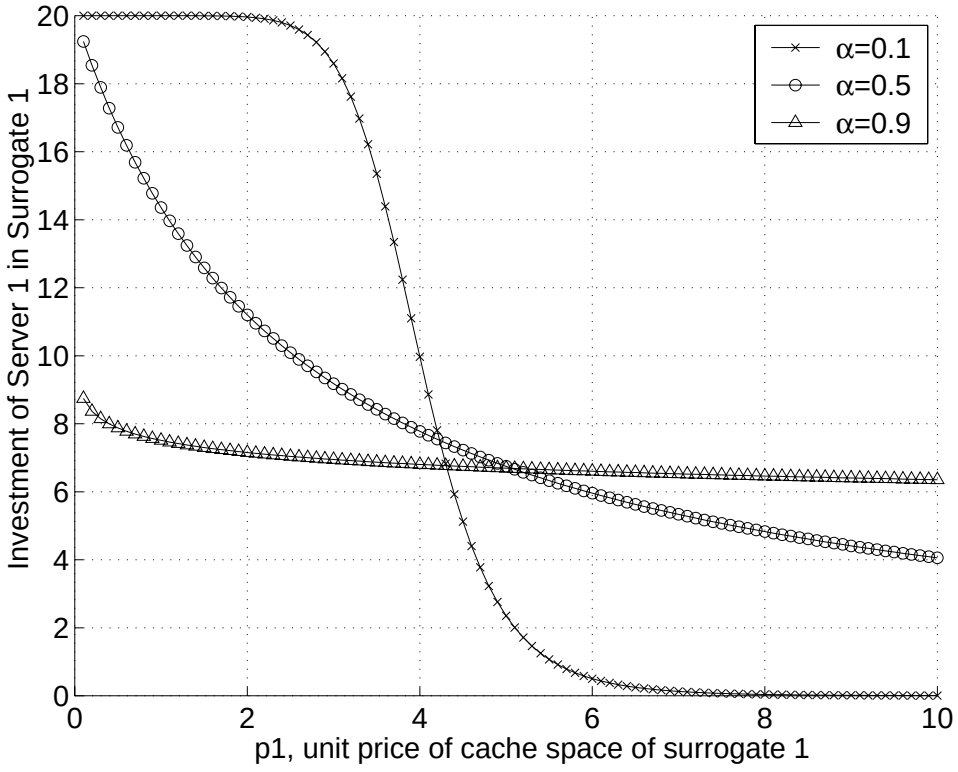

Figure 3.3: Elasticity of publisher investment when the cache size is infinite. $\alpha_2 = 0.5$. $\beta_i^j = 1, \forall i, j$. $p_2 = 4$, and $p_3 = 7$.

This result has been analyzed for a two publishers, three surrogates system, where the total investment of each publisher is 20 cost units, and the sizes of the caches of each surrogate is the same at 10 storage units. Figure 3.3 depicts the investment of the publisher in a surrogate when a surrogate's unit cache space price is varied, while the prices of the remaining two surrogates' are kept the same. The arrival rates to each surrogate and the gain factors of each publisher-surrogate pair are the same. The analysis does not take into account the limited cache capacities of the surrogates. The investment in the surrogate decreases with the increasing price. However, more importantly the investment is quite dependent on the distribution of requests for the publisher's content. In fact $\alpha = 1$ represents a special case, where the publisher's investment in a surrogate is the same regardless of the price of the surrogate.

In Figure 3.4, the variation of total revenue generated by a surrogate publisher for varying surrogate prices is depicted. In this case, as the surrogate lowers its price, it receives higher investment from the publishers. However, lowering the price more than a certain price reduces the revenue, because the surrogate has a limited cache space and the publishers requests for more space cannot be satisfied.

3.3 Optimal Surrogate Strategy

We now consider the optimal pricing strategies of the surrogates maximizing their revenues. Let $\mathbf{p}^{-j} = (p_1, p_2, \ldots, p_{j-1}, p_{j+1}, \ldots, p_J)$ be the set of unit cache space prices of all the surrogates in the network except the jth one. We assume that there is no collaboration among the surrogates, and each surrogate tries to maximize their revenue non-cooperatively. The revenue of a surrogate j with price p_j is $r_j(p_j) = \sum_i x_i^j(p_j)p_j$, and the objective of the surrogate is to maximize $r_j(p_j)$ by determining the optimal price p_j.

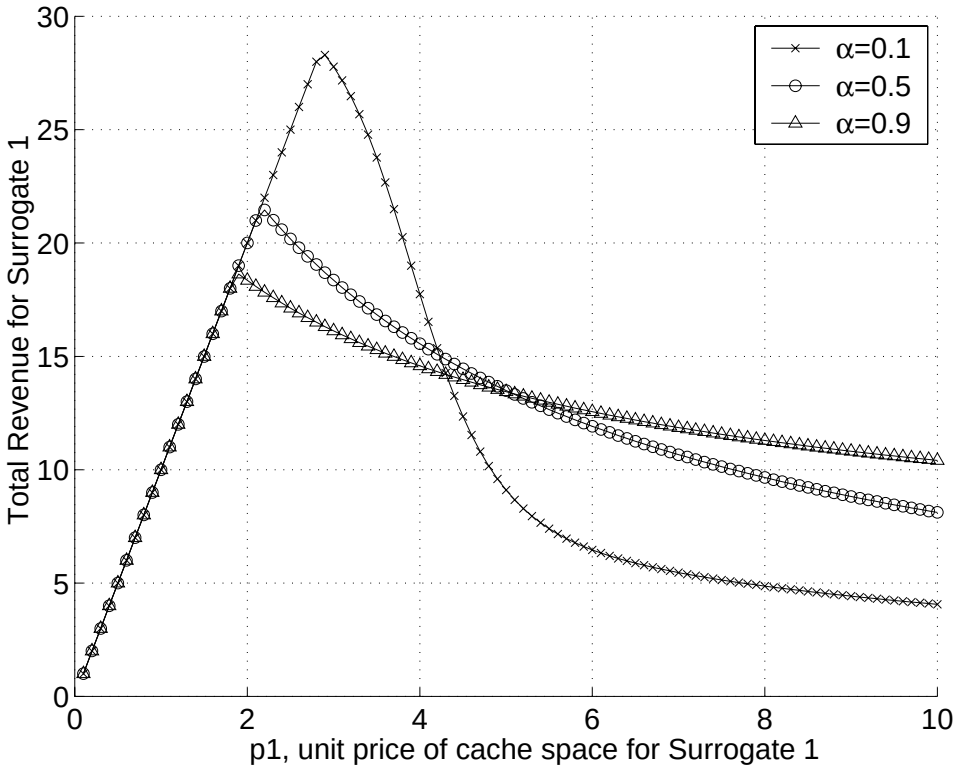

Figure 3.4: Total revenue of surrogate for varying prices. $\alpha_2 = 0.5$. $\beta_i^j = 1, \forall i, j$. $p_2 = 4$, $p_3 = 7$.

Lemma 2 *Surrogate j's best pricing strategy under a given fixed pricing policy $\mathbf{p}^{-j}$ is to set a price p_j that satisfies $\sum_{i=1}^{I} x_i^j(p_j) = C_j$, i.e., when the surrogate cache is completely allocated.*

Proof See appendix A. ■

This lemma suggests that the function $r_j(p_j) = \sum_i x_i^j(p_j)p_j$ achieves an interior maximum because it tends to zero both as p_j goes to zero and as p_j goes to infinity. Whether the function $r_j(p_j)$ is maximized at the price that completely allocates surrogate cache or at a higher price will depend on whether C_j is above or below this maximum. However, in many practical cases the cache capacity is much lower than the total information available in the network. Thus, the surrogate will be able to sell all of its capacity without setting a price that approaches zero.

3.4 Publisher-Surrogate Distribution Game

Until now, we discussed the optimal strategies of the publishers and the surrogates given that system is at a steady state. However, we have not discussed whether such a steady state exists. Notice that when a surrogate re-evaluates its pricing policy according to the pricing policies of the rival surrogates, the remaining surrogates will do the same. At each different pricing policy the publishers' optimal investments will be different as well.

In order to understand the behavior of the surrogates, we model the two-stage surrogate-publisher system as a non-cooperative game [41]. In this *publisher-surrogate distribution game*, $\Gamma(J, S, P)$, the players, J, are the surrogates, the strategy set S_j for a surrogate j is given by the surrogate's unit cache space price and the payoff function $P_j(s)$ of each surrogate j is given by the profit of the jth surrogate. This system is

similar to the Cournot oligopoly discussed in the economics literature. Assume that each surrogate has a fixed cost for its cache, but has no control over the size of the cache, i.e., the size of the cache is determined before the system implementation.

We first show that this game has a Nash equilibrium solution, where no surrogate has incentive to change its strategy unilaterally, since each surrogate maximizes its own individual payoff given the strategies of others.

Theorem 1 *The non-cooperative publisher-surrogate distribution game* $\Gamma(J, S, P)$ *has at least one Nash Equilibrium solution.*

Proof We first show that the strategy sets are convex and compact. The profit for surrogate j is $r_j(\mathbf{p}) - c_j$, where c_j is the cost of the surrogate j's cache. We will assume that there exists some price $\hat{p}_j$ at which demand for the cache space of surrogate j is zero regardless of the prices of other surrogates. Considering the revenue curve $r_j(\mathbf{p})$, this is not a restricting assumption. Lemma 2 suggests that the optimal surrogate price tends to zero when we deviate from the suggested optimal point. As an example, consider Figure 3.4, in which the revenue of surrogate j increases until a certain price p_j^* beyond which it decreases again. Then, we may limit the strategy set S_j to the interval $[0, \hat{p}_j]$, and still be able to cover the complete range of payoff function. Thus, the strategy set S_j is convex and compact.

The profit of each surrogate is bounded from below by zero and since the total investment of all publishers is limited, the profit can never exceed $\sum_i \mathcal{B}_i - c_j$. We assume that surrogate takes its' rivals actions as given, supposes they will remain constant, and chooses its own best course of action accordingly. This assumption is called *Cournot behavioral assumption* [41]. The payoff function under this assumption is given by $r_j(\mathbf{p})$. In Lemma 2, we have shown that there is a unique *best reply function*,

$R_j(\mathbf{p}) = \text{argmax}_{p_j}\{r_j(\mathbf{p})\}$ for surrogate j, which is also continuous. Define a mapping $\mathbf{R}(\mathbf{s}) = (R_1(\mathbf{s}), \ldots, R_J(\mathbf{s}))$. By Brouwer's Theorem [41] $\mathbf{R}$ must have at least one fixed point $\mathbf{s}^* \in S$, where $\mathbf{s}^* = \mathbf{R}(\mathbf{s}^*)$. The definition of the best reply function $R_j(\mathbf{s})$ and Brouwer's Theorem tell us that $P_j(\mathbf{s}^*) \geq P_j(\mathbf{s}^*/t_j)$ for all $t_j \in S_j$ and $j = 1, \ldots, J$, where $\mathbf{s}^*/t_j$ is the strategy set when the jth surrogate's strategy is changed to t_j in the complete strategy set $\mathbf{s}^*$. This result is the definition of Nash equilibrium. ■

We have shown that there exists a set of equilibrium prices for such a system. The question that remains to be addressed is what the physical interpretation of such an equilibrium is.

Consider the publisher optimization problem (S) discussed in the previous section. In our system, every publisher tries to maximize its own benefit regardless of others subject to the availability of funds and caching space. The optimization problem of each publisher is related to each other with the constraint $\sum_i x_i^j \leq C_j$ for all surrogates, i.e. the publishers compete for the available cache resources. Thus, we can re-write the optimization problem for individual publishers as:

$$(S_i) \quad \max_{\{x_i^j\}_{j=1}^J} \quad U_i(x_i)$$

$$\text{subject to} \qquad (1) \qquad \sum_{j=1}^{J} x_i^j p_j \leq \mathcal{B}_i$$

$$(2) \qquad \sum_i x_i^j \leq C_j, \ j = 1, \ldots, J.$$

Theorem 2 *When there is a unique equilibrium for the publisher-surrogate distribution game, $\Gamma(J, S, P)$, the equilibrium prices solve the optimization problem S_i for all publishers $i = 1, \ldots, I$, that is, the solution is globally* Pareto *optimum.*

Proof Assume that each surrogate uses the best reply function $R_j(\mathbf{p})$ to update

its price. The best price for surrogate j given the pricing policy $\mathbf{p}^{-j}$ is calculated from $\sum_i x_i^j = C_j$. At the equilibrium this condition is satisfied as well. Furthermore, the publishers calculate x_i^j as given by Lemma 1, which guarantees local optimality of the solution and the feasibility of the first condition in S_i. Uniqueness of the equilibrium guarantees that the feasible *locally* optimum solution is also the *global* optimum. Under these conditions, outcome of the publisher-surrogate game is the solution of $S_i, \forall i = 1, \ldots, I$. ■

Pareto optimality is the relevant criteria in a multi-objective problem setting such as ours. At the pareto optimum solution, one can find no other feasible solution that increases some objectives while not decreasing at least another objective. Theorem 2 states that if the equilibrium is unique, then the outcome of the non-cooperative game is the optimal solution to the individual revenue maximization problems of the publishers. If there are multiple equilibria, however; the resulting cache allocations, $\mathbf{x}$, are only locally optimum. Unfortunately, there are often multiple Nash equilibria and depending on the initial prices as well as price update strategies the outcome of the game may not always be the optimal solution. In the following, we discuss a special case of the surrogate cache allocation problem, where the delay between each publisher-surrogate pair is the same and user request arrival rates to each surrogate and Zipf distributions for each publisher are identical. For this case, we determine the condition for which unique equilibrium exists.

Definition 1 *A mapping $T(p)$ is called contraction mapping, if $|T(p)-T(q)| \leq \lambda|p-q|$ for $\lambda < 1$ or if the mapping is differentiable $\partial T(p)/\partial p < 1$.*

It is easy to see that if the best-reply mapping $\mathbf{R}(\mathbf{p})$ is a *Contraction mapping*, then the equilibrium is unique [61].

Lemma 3 *When $\beta_i^j = \beta$, $\forall i, j$ and $\alpha_i = \alpha$, $\forall i$, then the best reply function $R_j(\mathbf{p})$ is a contraction mapping if the price vector* $\mathbf{p}$ *is limited to the region given by*

$$\frac{(1-\alpha)\sum_i \mathcal{B}_i/C_j p_l^{-1/\alpha}}{\left(\sum_{k\neq j} p_k^{1-1/\alpha}\right)^{1+\alpha}} < 1, \forall l.$$

Proof See appendix A. ■

Notice that the condition given in Lemma 3 is not a necessary but a sufficient condition which is probably more restrictive than the necessary condition. Let $\mathcal{R}_j$ be the region given by the above Lemma. Following Theorem gives the condition for optimality of the outcome of the game for the identical case.

Theorem 3 *If the range of the price vector,* $\mathbf{p}$*, is in the region $\cap_{j=1}^{J}\mathcal{R}_j$, the publisher-surrogate distribution game has a unique equilibrium.*

Proof The result follows from Lemma 3 and Theorem 1. ■

In order to understand the consequences of this theorem, we consider a system where there are only 3 surrogates. For demonstration purposes assume that $\alpha = 0.5$. Let $K_j = (1-\alpha)\sum_i \mathcal{B}_i/C_j$. Figure 3.5 depicts the uniqueness property for different values of K_j. The lines in the figure show the uniqueness condition satisfied with equality. The region enclosed by these three price lines correspond to the region described by Lemma 3, i.e. $\mathcal{R}_j$. We have different regions for different surrogates. These regions basically depend on K_j, which in turn depends on the publisher investments and the size of the surrogate's cache. We observe in Figure 3.5 that as K gets higher $\mathcal{R}_j$ gets smaller. Theorem 3 states that if the prices are confined into the intersection of these regions, we achieve a unique equilibrium. Notice that $\cap_{j=1}^{J}\mathcal{R}_j = \mathcal{R}_{j^0}$ for some $j^0 = arg\max_{1\leq j\leq J}\{K_j\}$.

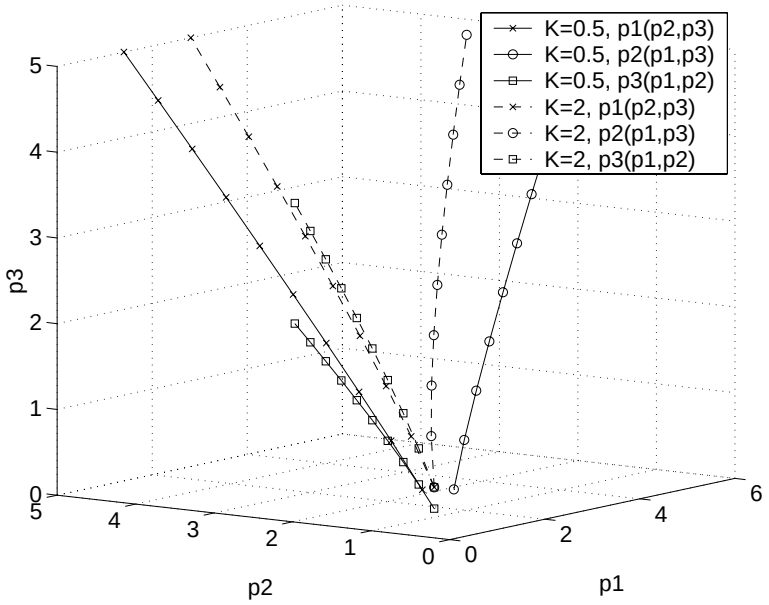

Figure 3.5: $\mathcal{R}_j$ for different surrogates.

We need to determine under what conditions the optimal surrogate prices lie in this region. Figure 3.6 and 3.7 depicts the variation of optimal surrogate prices given that the rest of the surrogates are required to select prices in the *uniqueness region*, i.e. $\cap_{j=1}^{J}\mathcal{R}_j$. We varied the boundaries of the uniqueness region by changing K. Notice that the number above each subplot corresponds to this K value. We noticed that for low values of K, the optimal surrogate prices reside inside the uniqueness region, while for high values of K the optimal prices tend to diverge from this region. Notice that low K values mean the total investment is not much higher than individual surrogate's cache size. Thus, if the size of the surrogate caches are not very large and/or the publishers are reluctant to invest high amounts in the surrogates, we expect the game to converge to a unique price equilibrium.

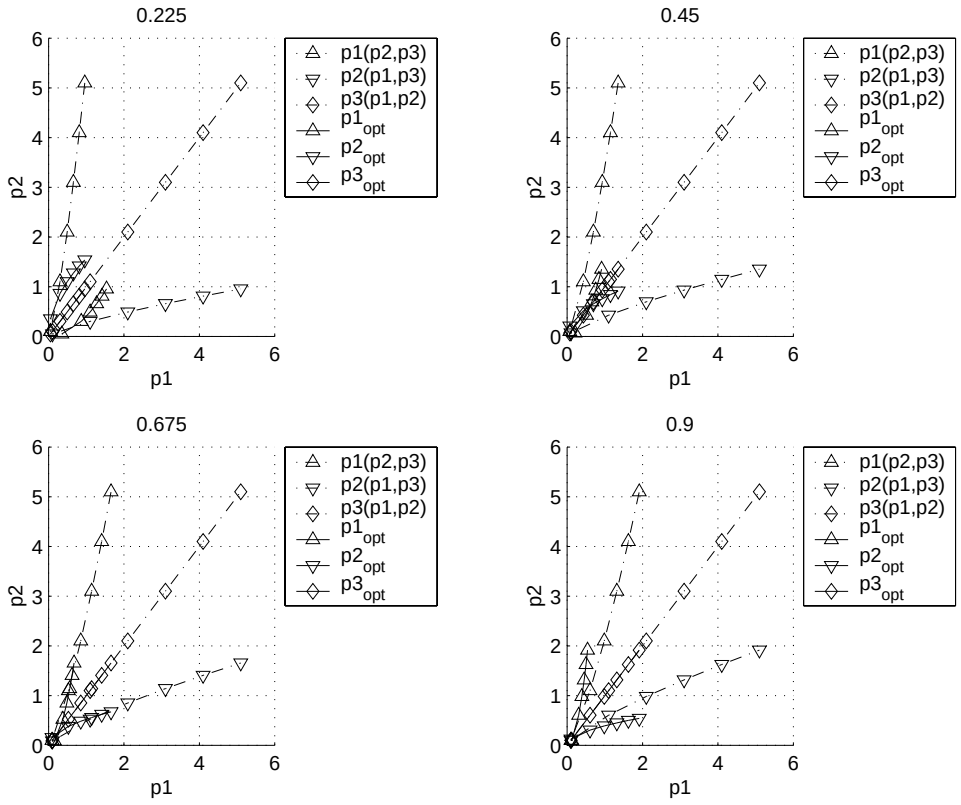

Figure 3.6: Variation of the optimal surrogate prices for different K.

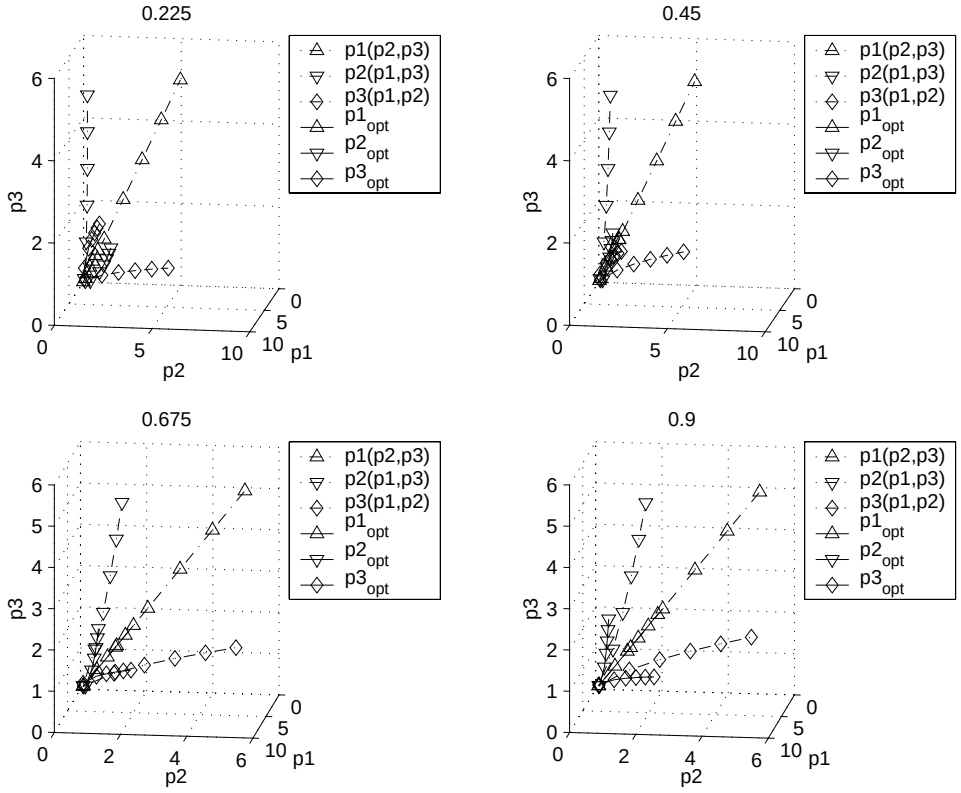

Figure 3.7: Variation of the optimal surrogate prices for different K.

3.5 Selection of Publisher Investments

So far, we assumed that publisher investments are fixed and known. We now discuss several possibilities for determining the investment amounts. Consider the following optimization problem (P). The objective of (P) is to maximize the total publisher utility given the limited cache sizes. We define the solution of (P) as the *system* optimum solution.

$$(P) \qquad \max_{\{x_i^j\}_{i,j}} \sum_i U_i(\mathbf{x}_i)$$

$$\text{subject to } \sum_i x_i^j \leq C_j, \qquad j = 1, 2, \ldots, J.$$

Define the Lagrangian for (P) as

$$\begin{aligned} L(\mathbf{x}, p) &= \sum_i U_i(\mathbf{x}_i) - \sum_j p_j \left(\sum_i x_i^j - C_j \right) \\ &= \sum_i (U_i(\mathbf{x}_i) - \sum_j p_j x_i^j) + \sum_j p_j C_j. \end{aligned} \tag{3.2}$$

Notice that the first term is separable in $\mathbf{x}_i$ and hence $\max_{\mathbf{x}} \sum_i (U_i(\mathbf{x}_i) - \sum_j p_j x_i^j) = \sum_i \max_{\mathbf{x}_i} (U_i(\mathbf{x}_i) - \sum_j p_j x_i^j)$. The objective function of the dual problem is

$$\begin{aligned} D(\mathbf{p}) &= \max_{\mathbf{x}} L(\mathbf{x}, p) \\ &= \sum_i \max_{\mathbf{x}_i} (U_i(\mathbf{x}_i) - \sum_j p_j x_i^j) + \sum_j p_j C_j. \end{aligned} \tag{3.3}$$

Thus, each publisher's optimization problem is separate from the other given the Lagrangian constant (shadow price) $\mathbf{p}$. By using similar arguments as those given in [62], one can easily show that the publishers can be induced to solve this optimization

problem in a distributed fashion by surrogate charging, and dual optimal prices $\mathbf{p}^*$ exist and $\mathbf{x}(\mathbf{p}^*)$ is primal dual as well. The dual problem $D(\mathbf{p})$ can be solved by using gradient projection method [9], where the resource prices are adjusted in opposite direction to the gradient $\nabla D(p)$.

$$p_l(t+1) = \max\{0, p_l(t) - \eta \frac{\partial D}{\partial p_l}(p(t))\}, \tag{3.4}$$

where η is the step size and $\frac{\partial D}{\partial p_l} = C_j - \sum_i x_i^j(\mathbf{p})$. This way, in each iteration publishers individually determine their optimal x_i^j and communicate their results to the surrogates. Surrogates then update their prices p_j according to Eq.(3.4) and communicate the new prices to the publishers and the cycle repeats.

Now, consider a publisher-surrogate game where a publisher selects an investment maximizing the "net" publisher benefit. That is, the difference of the additional average benefit generated from using surrogates and the cost of using them is maximized. Let T be the duration of time a publisher can rent the caching space from a surrogate. Then for a given set of surrogate prices $\mathbf{p}$ the publisher optimization problem can be given as,

$$\begin{aligned} \max_{\{x_i^j\}, \mathcal{B}_i} \quad & U_i(\mathbf{x}_i) - \frac{\sum_j x_i^j p_j}{T} \\ \text{subject to} \quad & \sum_j x_i^j p_j \leq \mathcal{B}_i. \end{aligned} \tag{3.5}$$

It is easy to see that the solution to this problem is obtained when the constraint is active i.e. satisfied with an equality. The solution to the optimization problem described in Eq. (3.5) is

$$x_i^j = \left(\frac{\beta_i^j (1-\alpha_i)}{p_j/T} \right)^{1/\alpha_i}, \tag{3.6}$$

$$\mathcal{B}_i \;=\; \sum_j \left(\frac{\beta_i^j (1-\alpha_i)}{p_j/T} \right)^{1/\alpha_i} p_j. \tag{3.7}$$

Similar to the case where the publishers have fixed investments, let $r_j(p_j)$ denote the total revenue generated by the surrogate j when it sets a price p_j. One can easily show that since $\frac{\partial r_j}{\partial p_j} = \sum_i (\beta_i^j (1-\alpha_i) T)^{1/\alpha_i} p_j^{-1/\alpha_i} < 0$ when $\sum_i x_i^j(p_j) \leq C_j$, the optimal surrogate pricing strategy is the one that results in full utilization of the cache resources. Thus, Theorem 1 still applies and the solution of this surrogate-publisher game has an equilibrium.

It is easy to see that the dual system optimization problem $D(\mathbf{p})$ is the same as our market interpretation. The two solution methods should give the same result as long as the investment for each server $\mathcal{B}_i$ is sufficiently large. This is due to the fact that the best reply function in the publisher-surrogate game requires the demand to be equal to the supply. However, $\mathcal{B}_i$ is usually a decision variable. Notice that a server may be individually better off by investing less than the amount dictated by the system optimum solution. Thus, a system optimum solution can only be achieved by the distributed publisher-surrogate game by enforcing the publishers to pay the charges associated with the system optimum solution.

Without such enforcement, we encounter a different optimization problem where the investment amount is also a design parameter. In the resulting non-cooperative game, publishers select their investment amounts to maximize their *profits*. We developed a suboptimal investment strategy for the publishers by assuming that the change in the equilibrium prices is *small* when the change in the investments is small. For a given investment B_i, the optimal cache allocation is given by $x_{ij}(B_i) = \frac{\beta_{ij}^2/p_j^2}{\sum_k \beta_{ik}^2/p_k} B_i$. Assume that $\{p_j\}$ are the set of prices at the equilibrium. As for the key assumption of

our derivation, we assume that for a small change in investment amounts, the change in the equilibrium prices is small. Specifically, in the vicinity of the set of investment amounts B_i, $\sum_k \beta_{ik}^2/p_k$ can be considered as *constant*. This assumption is verified through numerical experiments. Assuming that this condition is satisfied, the optimal investment amount maximizing the net benefit is calculated. Let $\gamma_{ij} = \frac{\beta_{ij}^2/p_j^2}{\sum_k \beta_{ik}^2/p_k}$, and $U_i = \sqrt{B_i}$. Then it turns out that the optimal publisher investment is the solution to the following nonlinear equations:

$$1/T - 1/2U_i \sum_j \beta_{ij}\gamma_{ij}^{3/2} + 1/2U_i^3 \sum_j \frac{\beta_{ij}\gamma_{ij}}{C_j} = 0, \tag{3.8}$$

and $\sum_i \gamma_{ij} B_i = C_j, \forall j$. Further details can be found in appendix B. The performance of this method is evaluated numerically and the results are given in Section 3.7.

3.6 An Optimization-based Resource Pricing Policy

The results given in the previous sections suggest that we may use a price-directed market-based distributed algorithm for solving the two-stage publisher-surrogate cache resource allocation problem. We consider the following algorithm for this purpose:

Resource Allocation Algorithm

1. Surrogates announce a set of initial prices $\mathbf{p}^{(0)} = (p_1^{(0)}, p_2^{(0)}, \ldots, p_J^{(0)})$.

2. At iteration k, each publisher i calculates its optimal cache demand for surrogate j, $x_i^{j(k)}$ as given in Lemma 1. Forward these demands to the surrogates.

3. At iteration k, each surrogate j updates its price according to the publisher

demands.

$$p_j^{(k+1)} = \max\{\epsilon, p_j^{(k)} + \gamma(x^j(\mathbf{p}^{(k)}) - C_j)\},$$

where $x^j = \sum_j x_i^j$ and γ is the step size. Let $\epsilon > 0$ be a sufficiently small constant preventing prices to approach zero. Thus, if the total demand $\sum_i x_i^{j(k)}$ is greater than the cache capacity C_j, then the new price $p_j^{(k+1)}$ is increased, otherwise it is decreased.

Announce the new prices $\mathbf{p}^{(k)}$ to the publishers.

In this model, the system operates as follows: an initial set of prices is announced to the publishers. The publishers determine their resource (cache) demands according to these prices as well as the request rates, and the observed delays from the surrogates. The publishers request these resources from the surrogates. Prices are then iteratively changed to accommodate the demands for resources until the total demand equals to the total amount of resources available.

The idea of using the optimality conditions to develop algorithms that solve an optimization problem is not new. For example Kelly et al. [48], Yaiche et al. [85] and Low et al. [62] have used this approach as well. These papers considered the optimal flow control in broadband networks. While Kelly proposed a continuous time algorithm in [48], Yaiche et al. and Low suggested discrete time algorithms. In each case the algorithms have different properties associated with the optimization problem they are designed for.

We next show that our algorithm indeed converges to the Nash equilibrium solution. Let V be a real valued function defined on R^J as follows:

$$V(\mathbf{p}) = \sum_j (p_j - \hat{p}_j)^2/\gamma, \tag{3.9}$$

where $\hat{p}_j$ is the set of equilibrium prices satisfying $x^j(\hat{p}_j) = C_j$. We consider V as a candidate Lyapunov function associated with the subsequence $\{\mathbf{p}^{(k)}\}$. Notice that V is convex, bounded and differentiable.

Theorem 4 *Let $\{\mathbf{p}^{(k)}\}$ be a sequence generated by the above algorithm for an arbitrary initial value $\mathbf{p}^{(0)} \in \mathbf{R}^J$ and $0 < \gamma < 1$. Then $\{\mathbf{p}^{(k)}\}$ converges to $\{\alpha\hat{\mathbf{p}}\}$, $\alpha \in \mathbf{R}$.*

Proof Define $E_j(p^{(k)}) = x^j(p^{(k)}) - C_j$. We know that $\frac{\partial V}{\partial p_j} = 2/\gamma(p_j - \hat{p}_j)$. It easy to see that ∇V is Lipschitz continuous, that is:

$$\|\nabla V(p) - \nabla V(q)\| \leq L\|p - q\|,$$

for $L = 1$. Thus, by the descent lemma [8, Appendix A.24], we have the following:

$$\begin{aligned} V(p^{(k+1)}) &\leq V(p^{(k)}) + \gamma E(p^{(k)})\nabla V(p^{(k)}) + \|\gamma E(p)\|^2 \\ &= V(p^{(k)}) + 2\sum_j (p_j^{(k)} - \hat{p}_j)E_j(p^{(k)}) + \gamma^2 \sum_j E_j^2(p^{(k)}) \\ &= V(p^{(k)}) + \sum_j \left[2(p_j^{(k)} - \hat{p}_j) + \gamma^2 E_j(p^{(k)})\right] E_j(p^{(k)}) \end{aligned} \tag{3.10}$$

Notice that from the definition of the algorithm, the following is true:

$$p_j^{(k+1)} - p_j^{(k)} = \gamma E_j(p^{(k)}), \tag{3.11}$$

for $p_j^{(k+1)} > \epsilon$. Consider the following two cases:

- $p_j^{(k)} \leq \hat{p}_j$

  For this case from Lemma 2 we know that $E_j > 0$. By definition of the algorithm $p_j^{(k)} \leq p_j^{(k+1)} \leq \hat{p}_j$. Then from Eq.(3.11),

$$\begin{aligned} p_j^{(k)} - \hat{p}_j &\leq p_j^{(k)} - p_j^{(k+1)} \\ &= -\gamma E_j(p^{(k)}). \end{aligned}$$

Now, consider the second term on the right hand side of Eq.(3.10) and the above result,

$$\begin{aligned} 2(p_j^{(k)} - \hat{p}_j) + \gamma^2 E_j(p^{(k)}) &\leq (p_j^{(k)} - \hat{p}_j) - \gamma E_j(p^{(k)}) + \gamma^2 E_j(p^{(k)}) \\ &= (p_j^{(k)} - \hat{p}_j) - \gamma(1-\gamma) E_j(p^{(k)}) \\ &\leq p_j^{(k)} - \hat{p}_j, \end{aligned}$$

where the last inequality follows from the assumption that $0 < \gamma < 1$. Thus,

$$V(p^{(k+1)}) \leq V(p^{(k)}) + \sum_j (p_j^{(k)} - \hat{p}_j) E_j(p^{(k)}),$$

and since the second term on the right hand side of above inequality is negative, $V(p^{(k+1)}) \leq V(p^{(k)})$.

- $p_j^{(k)} > \hat{p}_j$

  For this case from Lemma 2 we know that $E_j > 0$. By definition of the algorithm $p_j^{(k)} \geq p_j^{(k+1)} \geq \hat{p}_j$. Then from Eq.(3.11),

$$\begin{aligned} p_j^{(k)} - \hat{p}_j &\geq p_j^{(k+1)} - \hat{p}_j \\ &= -\gamma E_j(p^{(k)}). \end{aligned}$$

  Similar to the previous case, consider the second term on the right hand side of Eq.(3.10) and the above result,

$$\begin{aligned} 2(p_j^{(k)} - \hat{p}_j) + \gamma^2 E_j(p^{(k)}) &\geq (p_j^{(k)} - \hat{p}_j) - (1-\gamma)(p_j^{(k)} - \hat{p}_j) \\ &\geq p_j^{(k)} - \hat{p}_j, \end{aligned}$$

where the last inequality follows from the assumption that $0 < \gamma < 1$. Thus,

$$V(p^{(k+1)}) \leq V(p^{(k)}) + \sum_j (p_j^{(k)} - \hat{p}_j) E_j(p^{(k)}),$$

since $E_j(p^{(k)}) < 0$. The second term on the right hand side is negative, which leads to $V(p^{(k+1)}) \leq V(p^{(k)})$.

Given that $V(p^{(k+1)}) \leq V(p^{(k)})$ for all k, the level set $\{p \in \mathbf{R}^J | V(p) \leq V(p^0)\}$ is compact. Therefore $\{p^{(k)}\}$ is bounded, and it has at least one limit point. The subsequence $\{p^{(k)}\}$ converges to the stationary point of V (i.e. $\nabla V(p) = 0$) since ∇V is continuous and $\lim_{k\to\infty} \nabla V(p^{(k)}) = 0$. However, notice that there are multiple stationary points, since $x_i^j(p)$ is a homogeneous function of first degree and thus $E_j(\alpha \mathbf{p}) = 0,\ j = 1, 2, \ldots, J$ for $\alpha \in \mathbf{R}$.

■

3.7 Numerical Analysis

We compare the outcome of our algorithm with current caching systems that store the most popular data in their cache. We model the current system for our purposes as follows: surrogate j allocates $\dfrac{\frac{\lambda_i^j}{\sum_k \lambda_i^k}}{\sum_k \frac{\lambda_k^j}{\sum_l \lambda_k^l}}$ portion of the caching space to publisher i. Notice that, in fact this algorithm is better than the current implementation, since it considers the importance of a particular surrogate for a publisher. That is, if the requests of publisher i are arriving mainly from the network serviced by the surrogate j, surrogate j gives more caching space to publisher i than the rest of publishers.

We compare the performance of the game-theoretical and conventional caching

algorithms according to the total publisher revenues. We again consider the two-publisher and three-surrogate system as illustrated in Figure 3.1. We compare the performances of two methods when the skewness of the system is increased. Specifically, we consider the case when one of the publishers receives more benefit from one of the surrogates while the other publisher receives more benefit from another surrogate. We expect each of the methods to find the appropriate allocation that maximizes the publisher benefits.

Unlike the previous studies, which were interested in the cache hit/miss ratio, we consider the total publisher benefit as the relevant comparison criterion. Notice that when the required quality of service is not satisfied, users stop browsing the web page, which leads to lower revenues for the publishers. In this work, we assumed that different publishers have different contents with varying levels of QoS and their benefit of delivering a content to a user is also different, and the objective of each publisher is to maximize its revenue.

In this analysis we assume that the total investment of each publisher and the cache sizes of all surrogates are the same. Let $\zeta_i^j = w_i(d_{ij})/\chi_i^{1-\alpha_i}$. Notice that $\beta_i^j = \lambda_i^j \zeta_i^j$. Figure 3.8 depicts the improvement of game-theoretic algorithm over the conventional caching solution. In this figure we compare the two algorithms for varying request arrival rates. When the request arrival rates are equal to 1, then the solution given by the game-theoretical algorithm and the conventional algorithm is the same. However, as the arrival rates become smaller or greater than 1, we observe that game-theoretical algorithm gives better performance.

In Figures 3.9 and 3.10 we consider the performance improvement when the arrival rates are fixed, but ζ is varied. From the definition of ζ_i^j and assuming that $w_i(\cdot)$ is the same for all publishers, one can note that by varying ζ_i^j, basically we change the

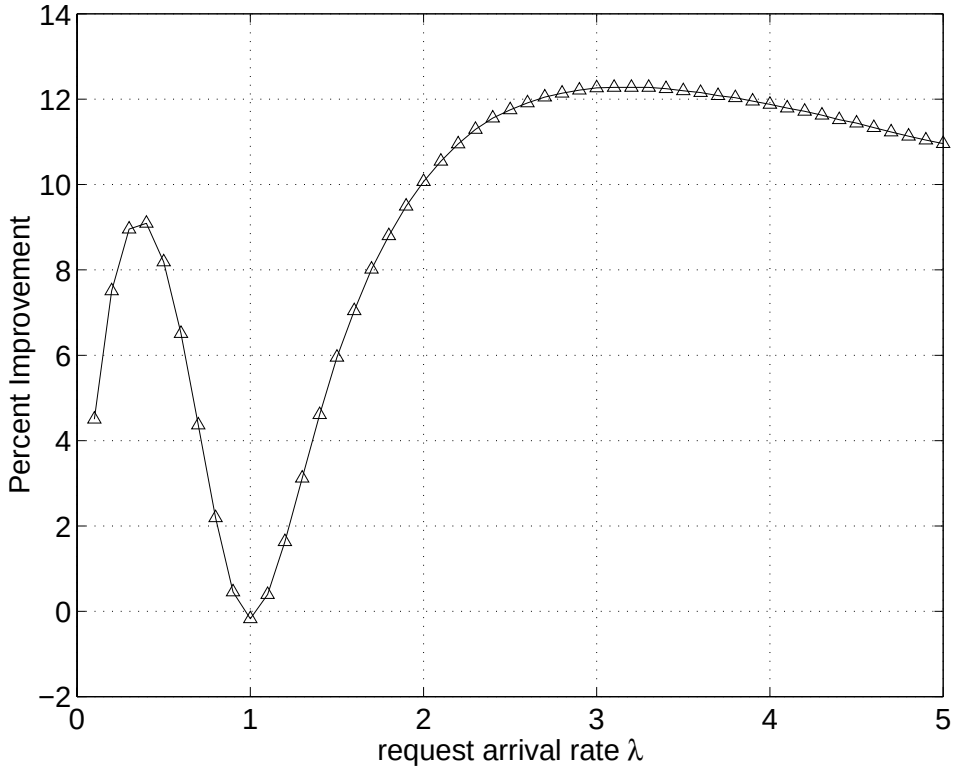

Figure 3.8: The effects of varying arrival rates. $\alpha_1 = \alpha_2 = 0.1$. $\zeta_i^j = 1, \forall i, j$. $\lambda_1^1 = \lambda_2^3 = \lambda$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$.

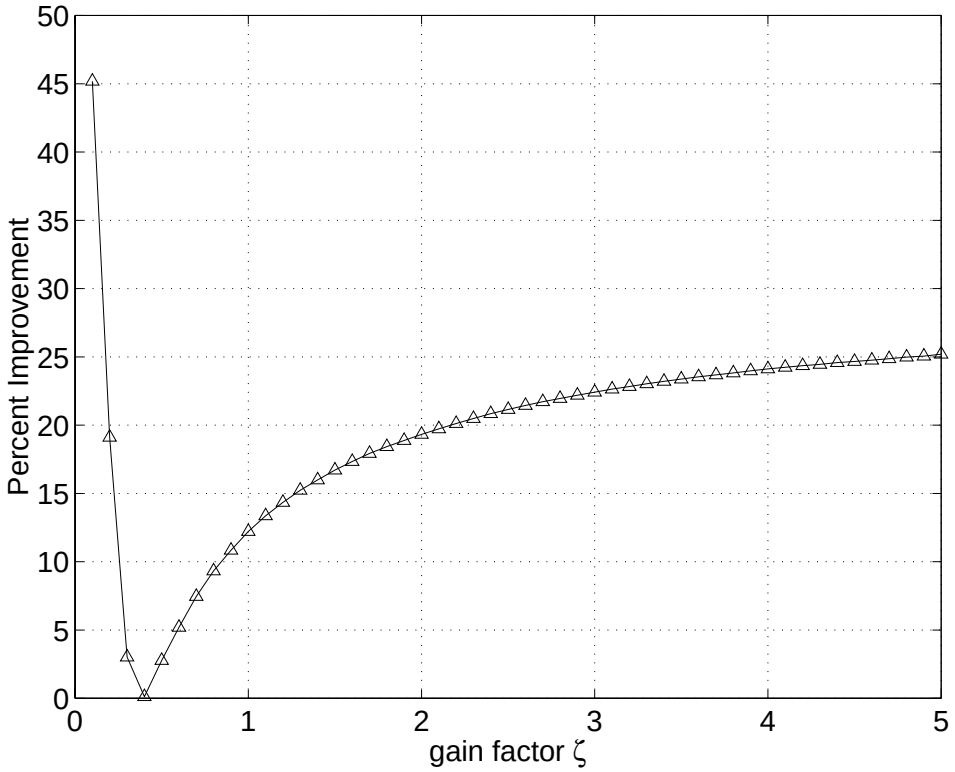

Figure 3.9: The effects of varying skewness in the network measured by varying the benefit received from several surrogates. $\alpha_1 = \alpha_2 = 0.1$. $\zeta_1^1 = \zeta_2^3 = \zeta$. $\zeta_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$. $\lambda_1^1 = \lambda_2^3 = 3$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$.

delay between surrogate j and publisher i. As illustrated in Figure 3.9, as the skewness of the system increases the performance of the game-theoretic algorithm gets better compared to the conventional algorithm. However, this improvement is larger for low values of ζ_i^j. This means that the surrogate does not have to be very close to the users. We may place surrogates at locations at somewhat greater distance and thus service more users for better efficiency.

Figure 3.10 depicts the performance when the publisher request characteristic is varied, i.e. when α_i parameter in the Zipf distribution of publisher i is varied. We observe that for larger values of α_i the improvement of the game-theoretic algorithm is smaller. This is reasonable considering that when $\alpha_i = 1$, the investment of a publisher is independent of the prices. In that case, the game-theoretic algorithm's solution reduces to the conventional algorithm's solution.

In Figure 3.11, we compare the performance of outcome of the game discussed in section 3.5. In Figure 3.11, *Alg.1* refers to the fixed investment method, where $\mathcal{B}_1 = \mathcal{B}_2 = 10$, and *Alg.2* refers to the optimal selection of the publisher investments leading to the maximum total system revenue according to the discussions in section 3.5. As discussed in the previous section, we expect that the total utility achieved by this game to be higher than the case when the investments are a priorly assigned. Indeed, Figure 3.11 verifies this intuition. It is interesting to note however that, while the system optimum investment amounts maximize the total system utility, they lead to lower utilities for some of the publishers compared to their utilities under the current caching methods. Furthermore, the investment amounts leading to the system optimum are usually very high, reducing any benefit received from the use of the caches in the first place. A short-sighted objective of maximizing total system utility thus seems non-implementable, when the system is formed by selfish non-cooperative

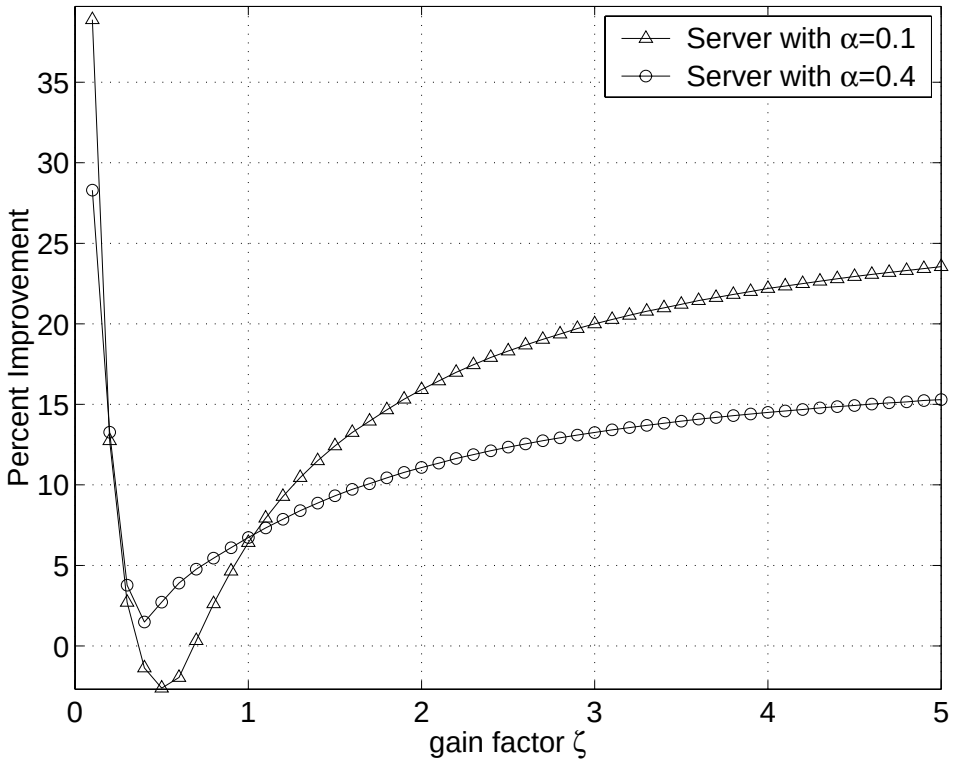

Figure 3.10: The effects of varying skewness when the user request distribution is different for the publishers. $\alpha_1 = 0.1$, $\alpha_2 = 0.4$. $\zeta_1^1 = \zeta_2^3 = \zeta$. $\zeta_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$. $\lambda_1^1 = \lambda_2^3 = 3$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$.

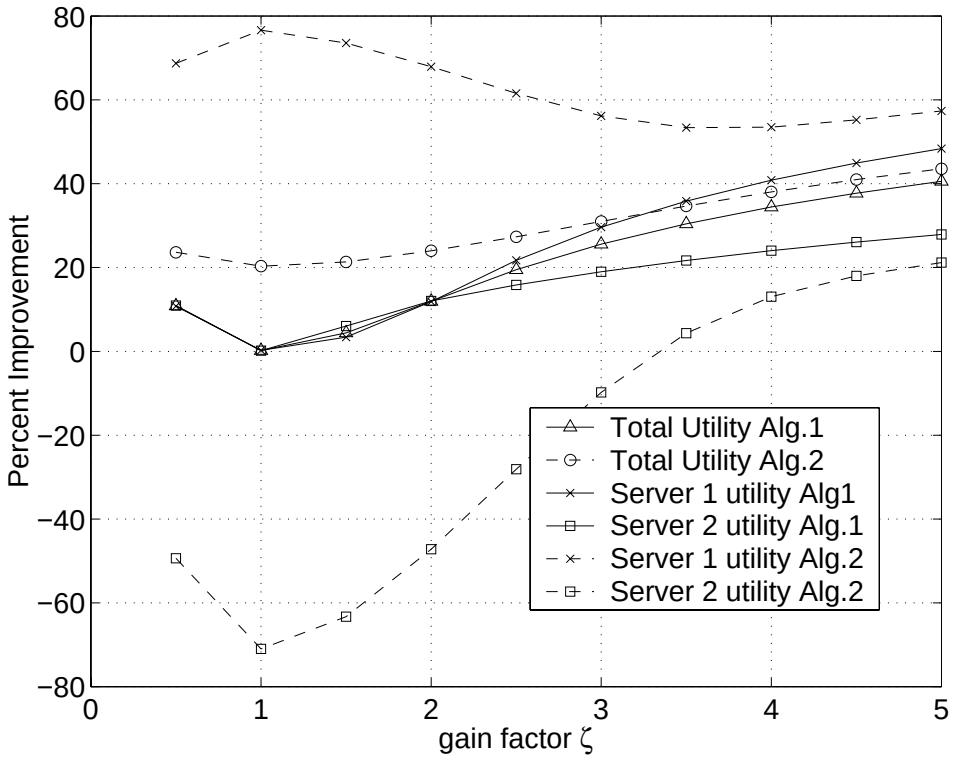

Figure 3.11: The illustration of system optimum solution does not increase all of the publishers benefits. $\alpha_1 = 0.1$, $\alpha_2 = 0.4$. $\zeta_1^1 = \zeta_2^3 = \zeta$. $\zeta_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$. $\lambda_1^1 = \lambda_2^1 = \lambda_1^3 = \lambda_2^3 = 3$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,1), (1,3), (2,3)$.

agents. We need to determine publisher investments that maximize the total system utility as much as possible, while keeping individual publisher benefits (or profits) higher than those under current solutions.

Thus we next consider the optimal investment strategies that maximize the publisher *profits*. Unfortunately, since the system is complex, we had to resort to an approximation. In this approximation, we assume that the change in the equilibrium prices is *small* when the change in the investments is small as well. We have verified this assumption through numerical studies. Given this assumption, we determined a sub-optimum investment strategy. The details and the derivation of our sub-optimal

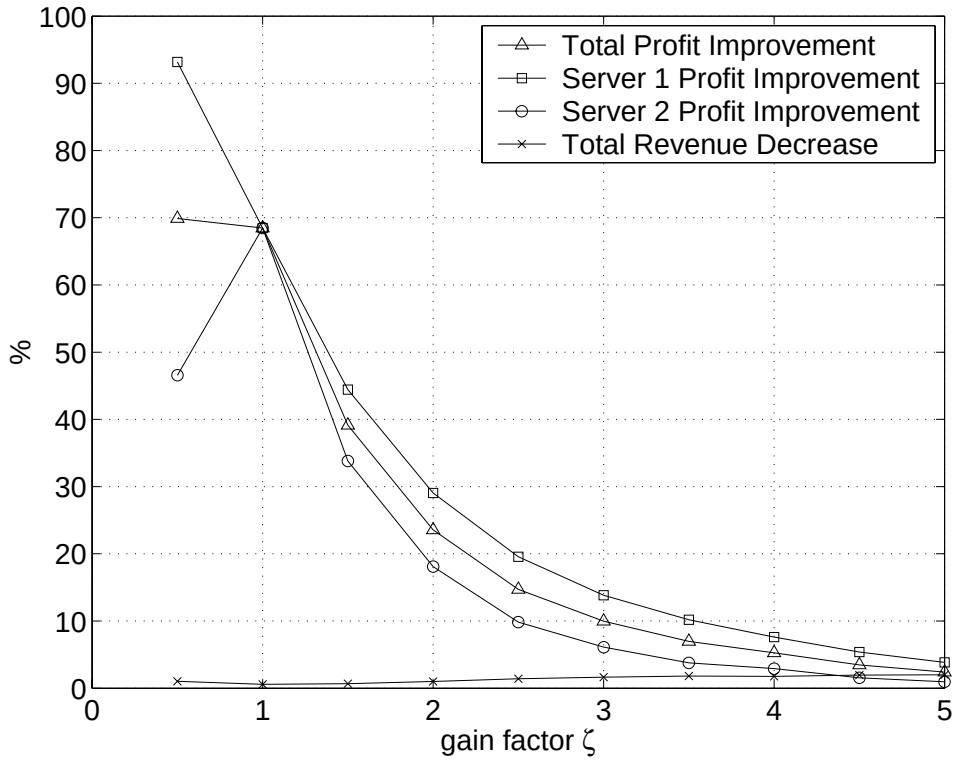

Figure 3.12: The improvements suggested by near-optimum investment strategy as compared with fixed investments $\mathcal{B}_1 = \mathcal{B}_2 = 10$. $\alpha_1 = \alpha_2 = 0.5$. $\zeta_1^1 = \zeta_2^3 = \zeta$. $\zeta_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$. $\lambda_1^1 = \lambda_2^1 = \lambda_1^3 = \lambda_2^3 = 3$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,1), (1,3), (2,3)$.

publisher investment strategy can be found in appendix B. Figure 3.12 depicts the improvement in the publisher profits compared to the a priori fixed investment case, where the fixed investment is 10. The system setup is similar to the previous numerical examples, and the improvement is considered for varying gain factors. We also depicted the reduction in the total publisher utility as compared to the optimal system utility. The cache rental period is 100 time units.

In Figure 3.13 we plot the change in improvement if the fixed investment was higher. In this case both publishers invest 15 units. In Figure 3.14 we show the change

in improvement if the skewness is increased further. We assumed that the gain factor of publisher 2 for the cache of surrogate 2 is also varied. In both of these figures, we observe that the improvement in the publisher profits is significant when the publishers are allowed to optimally vary their investments. Furthermore, it is interesting to note that the aggregate publisher utilities suggested by the non-cooperative method is very close to the system optimum. Notice that the system optimum required very high investments, which may diminish any benefit of using a caching system. Thus, we can say that the non-cooperative system can achieve high system efficiency without requiring any outside intervention.

We also noticed that the improvement in the profits reduce as the gain factors increase. The reason for this observation is that the profits of the publisher either with the fixed investments or with the optimal investments increase as the gain factors increase. However, for a given gain factor, the increase in the profit due to using the optimal investment is much lower than this increase. Thus, the rate of improvement decreases with the increase in the gain factor.

3.8 Conclusions

In this chapter, we analyzed a two-stage publisher-surrogate market-based resource allocation system. The objective was to deliver the contents to the users as rapidly as possible, so that users do not stop browsing the web sites. We assumed that the publisher contents are different and require varying levels of quality of service. When the required quality of service (e.g. latency) is not satisfied, the users start *bailing-out* according to some probability distribution. In this analysis, we showed that the publisher-surrogate game that models the system leads to an equilibrium. Under certain conditions we showed that this equilibrium is the optimal solution for

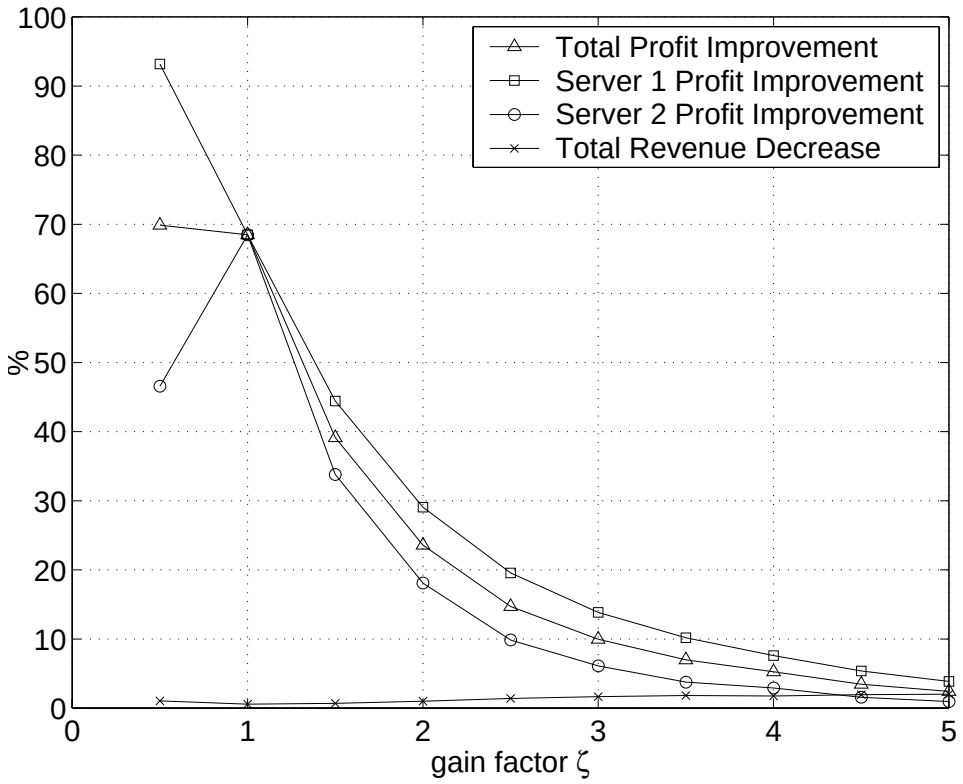

Figure 3.13: The improvement suggested by the near-optimum investment strategy as compared with fixed investments $\mathcal{B}_1 = \mathcal{B}_2 = 15$. $\alpha_1 = \alpha_2 = 0.5$. $\zeta_1^1 = \zeta_2^3 = \zeta$. $\zeta_i^j = 1, \forall (i,j) \neq (1,1), (2,3)$. $\lambda_1^1 = \lambda_2^1 = \lambda_1^3 = \lambda_2^3 = 3$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,1), (1,3), (2,3)$.

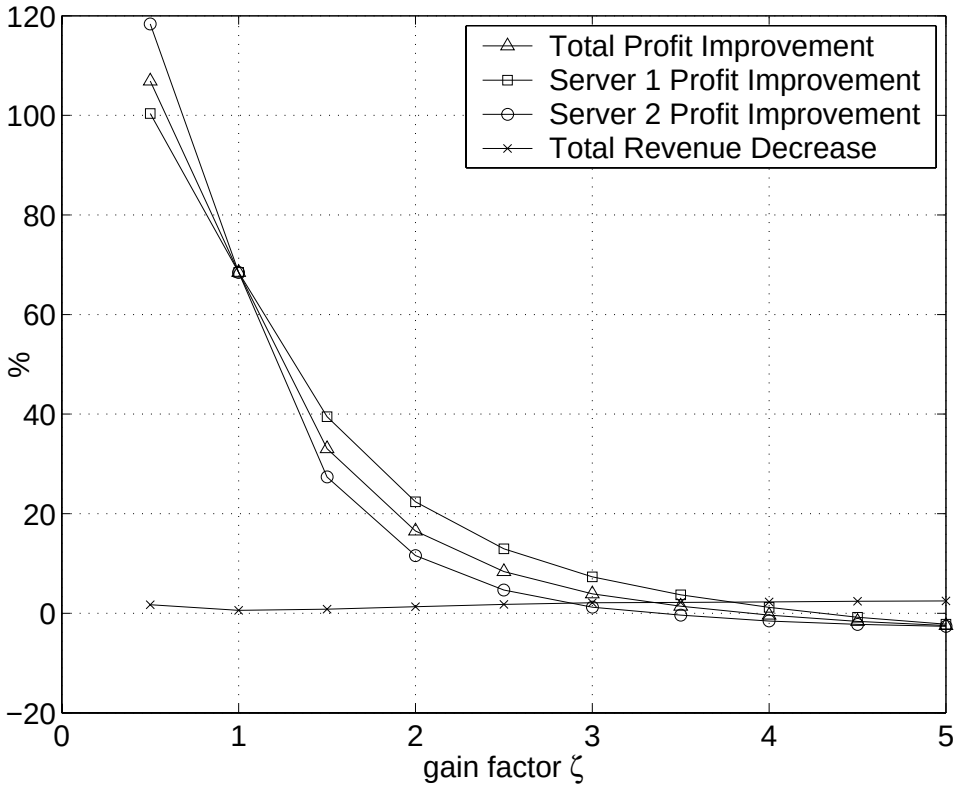

Figure 3.14: The improvement suggested by the sub-optimum investment strategy. $\alpha_1 = \alpha_2 = 0.5$. $\zeta_1^1 = \zeta_2^3 = \zeta_2^2 = \zeta$. $\zeta_i^j = 1, \forall (i,j) \neq (1,1), (2,3), (2,2)$. $\lambda_1^1 = \lambda_2^1 = \lambda_1^3 = \lambda_2^3 = 3$. $\lambda_i^j = 1, \forall (i,j) \neq (1,1), (2,1), (1,3), (2,3)$.

the non-cooperative resource allocation problem. The importance of our model is that it closely resembles the real-world situation, where the servers and users does not collaborate to achieve the system optimal solution. Instead every agent in the system tries to maximize their benefits without consideration of others. We also showed that the competition among surrogates leads to a solution that is better than the solution provided by conventional caching methods. Furthermore, we obtained the surrogate pricing strategies that maximize the total publisher benefit, which we consider as the system optimum. We noticed that the non-cooperative publisher-surrogate game, where the publishers also vary their investment to maximize their profits, leads to an equilibrium solution which is not far from system optimum and improves the publisher profits significantly. This result strongly suggests that the non-cooperative structure of the Internet is sufficient for efficient use of the network resources.

Chapter 4

User Request Routing

In this chapter, we focus on request-routing sub-system. We assume that the publishers disseminate their content to appropriate surrogates in one or more CDNs with the coordination of distribution sub-systems. CDNs peer with each other so that a client request may be diverted to the surrogate of another CDN, whenever it is optimal to do so. Peered CDNs keep information about the content and the network state of their own and each others' surrogates. We assume that the RRS has the information on the content and the network state of every surrogate in-house or peered (by bloom-filters, delta updates, periodical polling, etc). According to this information, the client requests are directed to the most appropriate surrogate. CDNs offer the publishers a SLA, which specifies the average delay observed by the clients accessing publishers content. The propagation and processing delay between the surrogate and client network is known by the RRS. Surrogates charge a fee for each client request served. This price can be considered as the shadow-price reflecting the current load on the servers. The objective of the RRS is to keep the average delay experienced by the clients of a publisher below a negotiated value while minimizing the total service cost.

Previous work in this field include the work-in-progress specifying the architecture

and issues in *Content Distribution Inter-networking* by the Internet Working Groups and the Content Alliance [25], [34], [68], [2], [6] and [14]. Recently Biliris et al. [11] considered the issues in request routing in peered CDNs. In their work, the authors considered methods for forwarding the client requests to the CDNs/surrogates. They suggested a DNS re-routing based scheme that they call Intelligent DNS (IDNS). IDNS dynamically updates the routing table for DNS engine in real-time according to the observed network state. The routing decision tries to balance the load in the surrogates. However, the authors did not specify an optimal load balancing method. Our work focuses on this routing decision and we develop a dynamical algorithm that is both scalable and robust and that minimizes a measure of service cost.

We also discuss the surrogate-publisher *routing* game. In the previous chapter, we assumed that the routing decisions are given and the publishers determine their dissemination strategy depending on the surrogate caching space prices. In this chapter, we first investigate the game where the publishers (or, on their behalf, the CDNs) determine the routing of the user requests to the surrogates given the publisher dissemination strategy. We show that such a game has an equilibrium, but the existence of this equilibrium depends on the structure of the game. Specifically, we show that an equilibrium exists, if the publishers *auction* for the bandwidth resources (i.e. routing) instead of requesting a maximum delay bound. Later, we combine the dissemination and routing games and show that this combined game has an equilibrium.

The chapter is organized as follows: in the next section we develop the model for the request-routing sub-system and define the minimum cost delay-constrained application level routing problem. This approach differs from the previous section, where the publishers determine their SLA and optimize their resource allocations according to the cost of the service. We note that a different perspective to the problem appears,

when the users request a SLA and the CDN tries to deliver this SLA at a minimum cost to the publishers. They may deliver the SLA at a minimum cost to the publisher to prevent the competing CDNs taking away the business of the publishers. In section 4.2, we give a method to determine a tight lower bound to the problem, and define a near-optimal routing algorithm. In section 4.3, we evaluate the performance of this algorithm as compared to the lower bound. Section 4.4 gives a definition of the routing game in the same line of the previous chapter. This section discusses the existence of equilibrium of the routing game, while the following section investigates the existence of equilibrium for the *combined* dissemination and routing game.

4.1 System Model and Problem Formulation

A publisher and CDN makes a Service Level Agreement (SLA), which specifies the maximum average delay observed by the users accessing the publisher's content. The objective of the RRS is to minimize the total service cost (and thus to maximize the profit), while satisfying all the publishers' SLAs.

This is a very practical problem that has to be solved by the CDNs in order to maximize their profits, while honoring their contracts to the publishers. We may identify several cases, where the service costs appear: it may be the case that the CDN has no surrogates of its own and only acts as an intermediary between the publishers and the ISPs with caching space to rent. Then, the surrogate service fee refers to a real monetary cost for the CDN. Even though the CDN has it own surrogates, it may charge a fee to those user requests re-directed from the peer CDNs. Moreover, the number of user requests that a surrogate can serve simultaneously is limited and the surrogates may employ *nonlinear pricing schemes* to avoid congestion. Nonlinear pricing schemes have attracted much attention recently especially in the network flow

control for their implicit property of admission control [65], [79], [77]. Nonlinear pricing relies on the fact that the admission of a new user would reduce the available resources for the users that are already being serviced. This reduction in available resources per user then relates to an increase in the average service delay. The cost of the total increased service delay is reflected by the *shadow price* that a new user has to pay to get service given the current server load. Not all users value the resources the same. Thus, the users that does not want to pay the higher costs associated with the use of resources leave and the congestion is relieved. In our work, we minimize the total service cost given a source of surrogate fees. Such an objective is required by the CDNs to maximize their profits.

Figure 4.1 summarizes the operation of the system. We may consider the RRS as an enhanced version of a DNS server for our purposes. Users are located at mutually exclusive networks, which may correspond to the networks of the different local ISP. User requests are first directed to the corresponding CDN's request-routing subsystem via DNS resolution. Each CDN may serve one or more of the user networks. Usually different CDNs do not serve the same user network, unless they are peer CDNs. The RRS determines a suitable surrogate to serve the user request and informs the user of the location of this surrogate. For our purposes, we may envision in-house and peered surrogates as equivalent, as long as the RRS have sufficient information on peered surrogates such as their service price and distance to the client networks. The user having received the identification of the surrogate, requests the object from that surrogate.

We assume that the SLA is specified as the average delay experienced by the clients accessing a publisher. Let $\mathcal{D}_j$ denote the average delay bound requested by the publisher j. The content of the publisher consists of I units, which are distributed

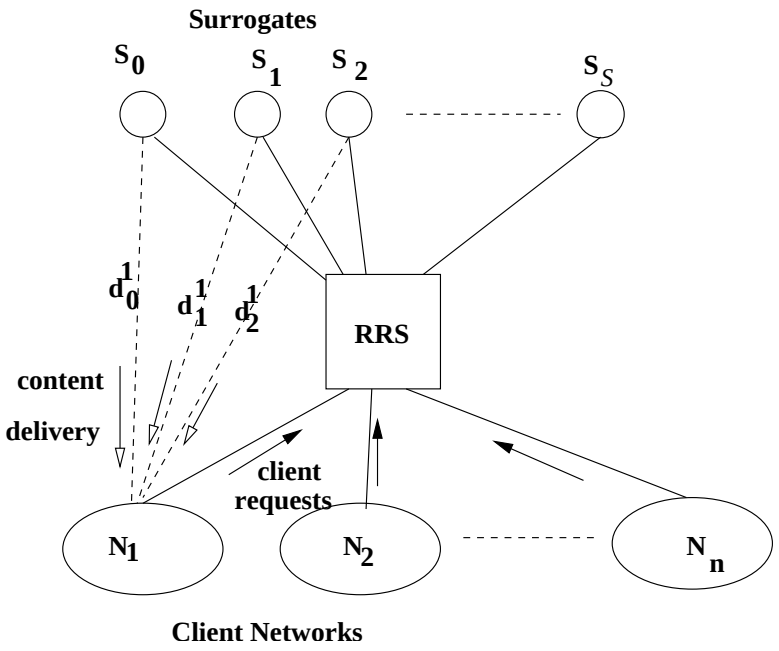

Figure 4.1: RRS Architecture.

among the set of $S_{ij} = \{s_0, s_1, \ldots\}$ surrogate servers. Let s_0 be the origin server for the requested object and contains all of the content. Notice that the number of objects in the Internet is quite high and it is not usually feasible to store the information of all the surrogates that store each of the objects. Instead, the RRS stores a summary of the surrogate cache content as suggested by such previous works as [26] and [73]. Thus, there is a possibility that some of the requested objects cannot be found in the surrogates. In that case, the surrogate requests the object from the origin server on behalf of the user. We consider the propagation and transmission delays together. When a user request from network n is routed to the surrogate s, the average retrieval delay experienced by the user is d_s^n, which includes both average propagation and transmission delays, including the delay required to fetch the object from the origin server when the object is not found in the surrogate cache. Assume that each content unit is of equal size. The request arrival rate for content i in publisher j from client

network n is given by λ_{ij}^n.The unit service price of surrogate s is p_s. Let $x_{ij}^n(s) = 0$ or 1 be the decision variable denoting the surrogate server serving the client requests from network n for the object i in publisher j.

The objective of the CDN is to minimize the total cost of service of all user requests, while satisfying the individual average delay bounds for the publishers. The following optimization problem (P) describes this objective:

$$(P) \quad \min_{\underline{x}} \sum_i \sum_j \sum_{s \in S_{ij}} \sum_n p_s \lambda_{ij}^n x_{ij}^n(s) \tag{4.1}$$

$$\text{s.t.} \sum_i \sum_{s \in S_{ij}} \sum_n \lambda_{ij}^n d_s^n x_{ij}^n(s) \leq \mathcal{D}_j, \; \forall j$$

$$\sum_{s \in S_{ij}} x_{ij}^n(s) = 1, \; \forall n, i, j.$$

This problem has close affinity to the delay-constrained routing problem investigated in the context of ATM networks, e.g. [87] and [74]. In the delay-constrained routing problem, clients are connected with each other over a multi-hop path. Each hop traversed in the network creates a delay and it is required that each connection's end-to-end delay is lower than a negotiated value. In our work, we consider application-level routing, so the client and the destination is always at single "hop[1]" distance. Contrary to the delay-constrained routing in ATM networks, the user connections are coupled in the sense that overall average delay experienced by the clients of a publisher should be lower than a negotiated delay. (P) is an integer programming problem and it was shown that the similar delay-constrained routing problem is NP-complete [17].

[1]Notice that user *packets* may traverse multiple network elements/links to reach the server.

4.2 Lower Bound and a Near-Optimal Routing Algorithm

We determine a tight lower bound to (P) by solving the dual problem. We dualize the problem (P) with respect to the first constraint. Let $\alpha_j \geq 0$ be the Lagrangian constant.

$$(D) \quad Z_D(\alpha_j) = \min_{x_{ij}^n} \left\{ \sum_i \sum_j \sum_{s \in S_{ij}} \sum_n p_s \lambda_{ij}^n x_{ij}^n(s) \right.$$

$$\left. + \sum_j \alpha_j \sum_i \sum_{s \in S_{ij}} \sum_n \lambda_{ij}^n d_s^n x_{ij}^n(s) - \alpha_j \mathcal{D}_j \right\} \tag{4.2}$$

$$\text{s.t.} \sum_{s \in S-ij} x_{ij}^n(s) = 1, \, \forall n, i, j.$$

It is easy to see that the routing decision given by the solution to $Z_D(\alpha_j)$ is,

$$x_{ij}^n(s) = \begin{cases} 1 & \text{if } s = \arg\min_{s \in S_{ij}} \{p_s + \alpha_j d_s^n\}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

By weak duality we can determine a lower bound for (P) as the solution of $\max_{\alpha_j} Z_D(\alpha_j)$, since $Z_D(\alpha_j)$ is non-differentiable, we have to resort to one of the approximation methods available in the literature. We consider the sub-gradient method (see e.g. [8] for details) to solve this optimization problem. The sub-gradient is $g_j^k = \sum_i \sum_n \sum_{s \in S_{ij}} \lambda_{ij}^n d_s^n x_{ij}^{n,k}(s) - \mathcal{D}_j$. The iterations are $\alpha_j^{k+1} = \alpha_j^k + s^k g_j^k$, where $\alpha_j^k \geq 0$ and s^k is the step size. The new iterate may not improve the dual cost for all values of the step size; however, if the step size is small enough, the distance of the current iterate to the optimal solution set is reduced.

We can determine a simple feasible solution to (P) as follows: Let $d_{min}^j = \frac{\mathcal{D}_j}{\sum_i \sum_n \lambda_{ij}^n}$. If $x_{ij}^n(s') = 1$ for s' such that $d_{s'}^n \leq d_{min}^j$, then the delay constraint is satisfied. Let

$S'(i,j,n) = \{s \in S_{ij} : d_s^n \leq d_{min}^j\}$ denote the set of surrogates satisfying this property for the corresponding object for the users in network n. Furthermore, if we choose s'' as $s'' = \arg\min\{p_s : s \in S'(i,j,n)\}$, that is for each client network and object such that $p_{s''}$ corresponds to the minimum price offered by the surrogates satisfying the delay constraint, then we get an upper bound for problem (P).

Notice that due to the strictness of the constraint $d_{s'}^n \leq d_{min}^j$, the gap between the delay bound and the total average delay under this scheme is quite large. Our near-optimal algorithm relies on reducing this gap intelligently, while reducing the total cost as much as possible. We first give our algorithm in detail:

- For each publisher j,
  - Determine

$$\begin{aligned} S'(i,j,n) &= \{s \in S_{ij} : d(s,n) \leq d_{min}^j\} \\ s(i,j,n) &= \arg\min\{p_s : s \in S'(i,j,n)\} \\ S(i,n) &= \{s(i,j,n) : \forall i,n\} \end{aligned}$$

  - Determine $w^n(k,l) = \frac{p_l - p_k}{d_k^n - d_l^n}$, $\forall n$ and $\forall k,l \in \cup_{i,j} S_{ij}$.
  - While $\sum_i \sum_n \lambda_{ij}^n d_{s(i,j,n)}^n \leq \mathcal{D}_j$ update the routing decisions as $(i^*, n^*, l^*) = \arg\max_{l \in S_{ij}, i, n}\{w^n(s,l) : s \in S(i,n), p_s - p_l > 0\}$, $s(i^*, j, n^*) = l^*$.

In this algorithm, initially starting from the routing scheme resulting in the upper bound, we reduce the total cost iteratively by reducing the *slack* in the delay constraint. While increasing the total average delay to the delay bound, we choose the new surrogates that reduce the cost the most with minimum increase in the associated delay. This is performed by calculating for each user network the "benefit coefficient," $w^n(k,l)$, which corresponds to the reduction in total cost per unit increase in delay

by switching from the current surrogate assignment to a new one. In a *greedy* fashion we re-assign the user networks maximizing this coefficient while still maintaining the delay bound.

4.3 Numerical Analysis

Our objective in these numerical studies is to evaluate the performance of the proposed routing algorithms compared to the lower bound for different network conditions. First, we determine the lower bound by the sub-gradient method. The maximum number of iterations for the sub-gradient solution to the dual problem is 1000. The step size is $s^k = \delta \frac{Z_{up} - Z_D(\alpha_j^k)}{||g^k||^2}$, where Z_{up} is the total cost corresponding to the upper bound and δ is a constant. We initialize δ to be 2, but it is halved of its value when the objective value of the dual problem does not improve for 30 iterations.

The set-up corresponding to Figure 4.2 assumes that there are 10 publishers with 10 distinct objects. There are 10 client networks and 10 different surrogates. All surrogates contain all the objects published. The distance between a surrogate and a client network is given by $d_s^n = |s - n| + 1$, where $s, n = 1, 2, \ldots, 10$ are the indices of the corresponding surrogate and network respectively. The cost of service is given by $p_s = |5 - s| + 1$. This choice of distance matrix and service costs ensure heterogeneity in the network. The client requests for individual objects in the publishers content is distributed according to Zipf distribution with skewness parameter $\beta = 0.5$[2]. As shown in recent studies, Zipf distribution closely approximates the Web traffic [13]. The arrival rate for client requests to each publisher is assumed to be 1 request per second. The delay bound for each publisher is assumed to be the same. From Figure

[2]Probability that ith object is requested is, $Pr(i) = c/i^{\beta}$, where c is the normalization constant.

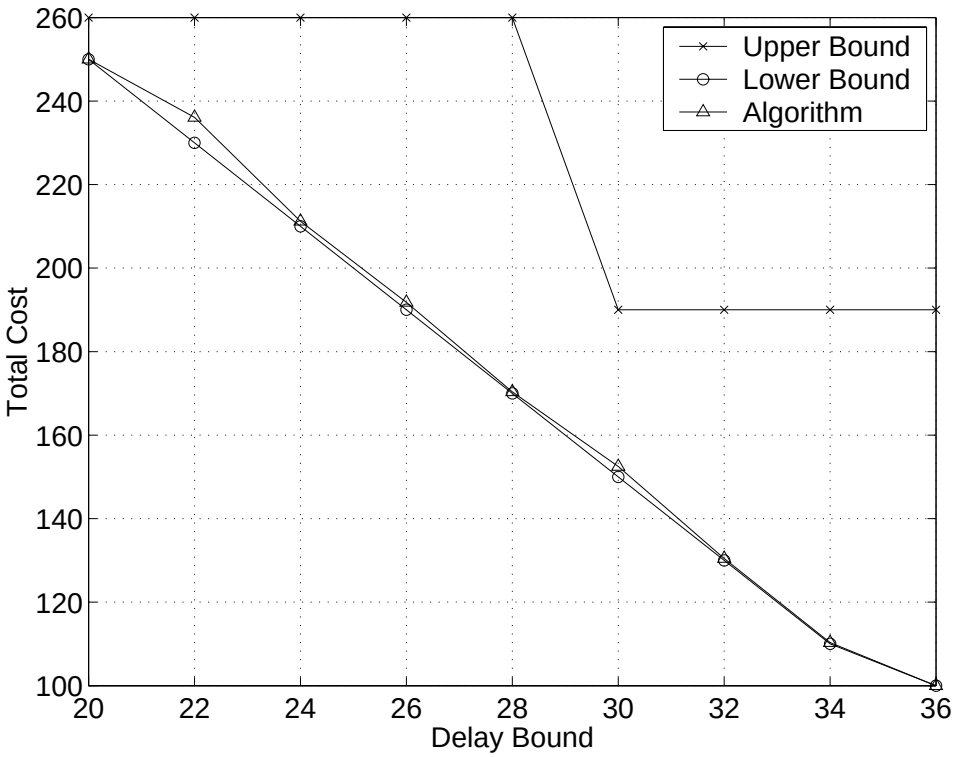

Figure 4.2: Total cost for uniform delay bound.

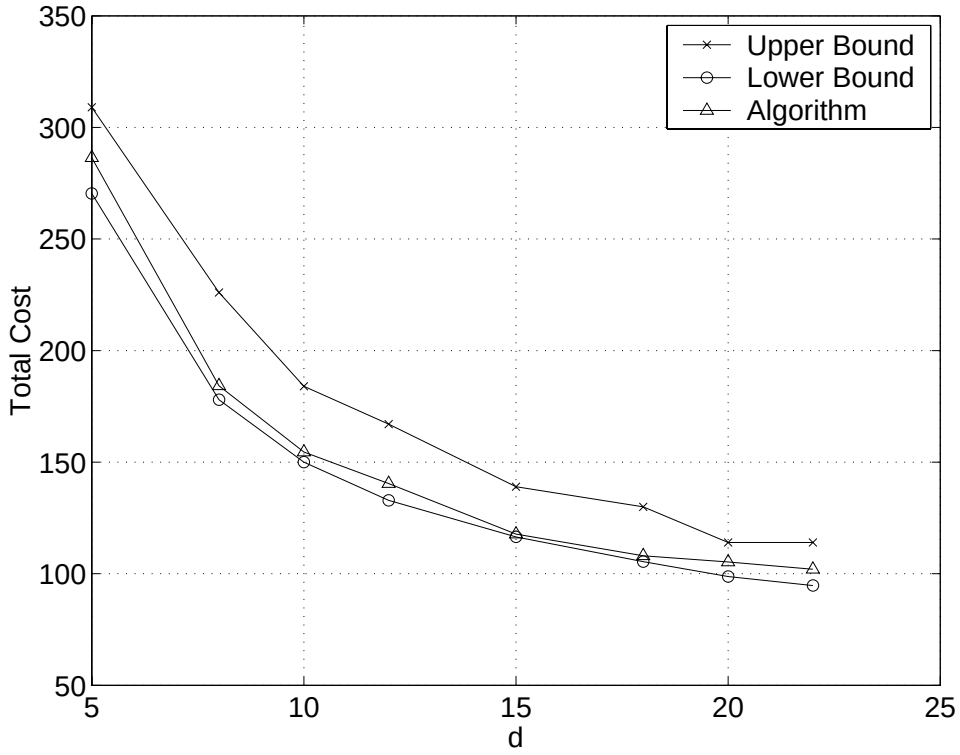

Figure 4.3: Total cost for non-uniform delay bound.

4.2 we observe that our algorithm can perform near-optimally compared to the lower bound. Furthermore, we observe that as we increase the number of objects published, the performance gets even better.

In Figure 4.3, we plot the effect of non-uniform delay bounds. We assume that users accessing the content of the publisher j should experience total average delay less than $(1+j/2)*d$, where d is varied in the simulation. We observe that our routing algorithm still provides performance close to the lower bound. We notice that when $d > 22$, our lower bound is no longer tight. From this point onwards the lower bound begins to correspond to un-realizable routing schemes. Another lower bound can be determined by routing all requests to the surrogate with the least price. In most cases this lower bound is not tight, but we may consider it when the lower bound given by

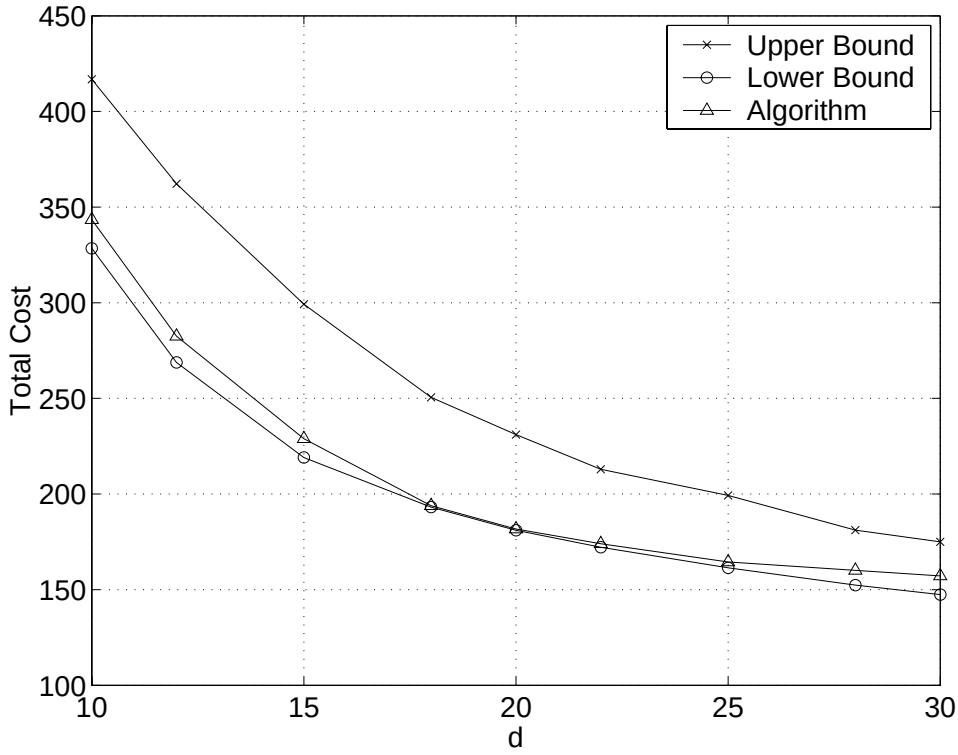

Figure 4.4: Total cost for non-uniform delay bound and non-uniform arrival rates.

the dual problem is no longer realizable.

In Figure 4.4, we show the effects of non-uniform arrival rates. While everything else is kept the same as in the previous analysis, we assume that the arrival rates are given as $\lambda_{i,j}^n = (j+n)Pr(i)$. Our proposed algorithm has a performance close to the lower bound.

4.4 Publisher-Surrogate Routing Game

While the RRS determines the best routing strategy minimizing the total cost, each surrogate updates its price p_s according to the prices of the other surrogates and the routing decision of the RRS. An important question is whether such a system has an equilibrium or not.

First we observe that when each publisher has a high number of objects, we may consider the total request arrival rate from network n to surrogate s for those objects in publisher j $\lambda_j^n(s) = \sum_i \lambda_{i,j}^n x_{i,j}^n(s)$ as a continuous function, since $\lambda_{i,j}^n << \lambda_j^n(s)$ for most of the objects. As $I \to \infty$, there always exists a routing strategy $x_{i,j}^n(s)$ that gives the desired $\lambda_j^n(s)$. Thus, as $I \to \infty$, the optimization problem (P) can be approximated by the following linear programming problem (P').

$$(P') \quad \min_{\lambda_j^n(s)} \sum_j \sum_{s \in \cup_i S_{ij}} \sum_n p_s \lambda_j^n(s) \tag{4.4}$$

$$\text{s.t.} \sum_{s \in \cup_i S_{ij}} \sum_n \lambda_j^n(s) d_{sn}^{tot}(j) \leq \mathcal{D}_j, \, \forall j$$

$$\sum_{s \in \cup_i S_{ij}} \lambda_j^n(s) = \sum_i \lambda_{i,j}^n, \, \forall n, j.$$

Let $d_{sn}^{tot}(j)$ denote the average delay experienced by the user requests from network n for the objects in the publisher j, when those requests are routed to surrogate s. Notice that $d_{sn}^{tot}(j) = \sum_i d_s^n x_{i,j}^n(s)$. $d_{sn}^{tot}(j)$ involves the average propagation delay between the surrogate and the network, the transmission delay associated with the surrogate and when the requested object is not found in the surrogate the average propagation delay to the publisher.

As discussed in the previous chapter for the allocation of caching resources, the surrogates still have the objective of maximizing their revenues, but this time by selling/renting their transmission resources. Notice that the surrogate optimization problem for the allocation of transmission resources is almost the same as the one for caching resources, and thus the surrogate revenue is similarly given as $r_s(p_s) = \min\{C_s^{tran}, \sum_i \sum_j \sum_n \lambda_j^n(s)\} p_s$, where $\lambda_j^n(s)$ is the solution to (P'). If $r_s(p_s)$ is continuous and concave, then it is known from Lemma 1 that the equilibrium exists.

In order for $r_s(p_s)$ to be continuous, $\lambda_j^n(s)$ must be continuous with respect to p_s. In the literature, the conditions for the continuity of the solution of a linear program, when the objective and the constraint set is changing linearly with respect to a parameter are rarely considered. Most recently, Korilis and Lazar [53] considered the continuity of the solution to a linear program in the context of network flow control. In this work, the users determine their rate by adjusting their window size according to the decisions of the users in the system. The users are assumed to be non-cooperative. The authors investigate the conditions under which an equilibrium exists. They show that the best-reply function is the solution to a linear program which depends on the window size of other users. As discussed in the previous chapter, an equilibrium exists when the best-reply function is continuous. In a more general context, sufficient conditions for the continuity of the set of solutions to a parameterized maximization problem are given by the "maximum theorem," due to Berge [7], which is one of the fundamental theorems employed in mathematical economics.

Theorem 5 Berge's maximum theorem [7]

Let X, Y be the subsets of two finite dimensional Euclidean spaces. Let $\mathcal{C} : X \to Y$ be a compact-valued correspondence and $f : X \times Y \to \mathbf{R}$ be a continuous function. $\Gamma : X \to Y$, $\Gamma(x) = \arg\max_{y \in \mathcal{C}(x)} f(x, y)$ is the solution to the optimization problem. Also, define $F : X \to \mathbf{R}$, $F(x) = \max_{y \in \mathcal{C}(x)} f(x, y)$. Then, if $\mathcal{C}$ is continuous at $x \in X$, then Γ is closed and upper semi-continuous at x, and F is continuous at x.

From Berge's maximum theorem, it is easy to see that the total cost to (P') is continuous and $\lambda_j^n(s)$ is upper semi-continuous. However, careful consideration of (P') hints that $\lambda_j^n(s)$ is not usually lower semi-continuous. Thus, $r_s(p_s)$ is not continuous either. In such a case, the equilibrium does not exist. Korilis and Lazar has developed

another condition for the continuity of the linear programs.

Theorem 6 Korilis-Lazar Continuity Principle [53]

Consider the following linear program continuously parameterized by $x \in X$*:* $\max_y\{c(x)y \mid A(x)y = b(x),\ y \geq 0\}$*. If there exists a compact set of solutions,* $W \subset \mathbf{R}^k$*, that is common for all dual programs with parameter* x*, then the solution to the linear program is continuous.*

The dual program of (P') is

$$(D') \quad \max_{\pi_j, \mu_{jn}} \sum_j \pi_j D_j + \sum_j \sum_n \mu_{jn} \sum_i \lambda_{i,j}^n \tag{4.5}$$

$$\text{s.t. } d_{sn}^{tot}(j)\pi_j + \mu_{jn} \leq p_s, \forall j, n, s.$$

A straightforward observation of the dual program suggests that the set of solutions to dual program does not have a common subset for all $p_s \in \mathbf{R}^+$, since the optimal solution to the dual program increases with increasing p_s. The abovementioned two theorems suggest that the linear program (P') is usually not continuous, so there is no equilibrium for the routing game.

Thus, we see that if the game is a solution to the linear program minimizing the total cost subject to a delay constraint, an equilibrium usually does not exit. However, an equilibrium exists when the publishers and surrogates play a game in which each try to maximize its own benefit similar to the distribution sub-problem discussed in the previous chapter. Consider (P_j'') as follows:

$$(P_j'') \quad \min_{\lambda} \sum_n \sum_s (\lambda_j^n(s))^{1-\alpha} d_{sn}^{tot}(j) \tag{4.6}$$

$$\text{s.t. } \sum_n \sum_s p_s \lambda_j^n(s) \leq \mathcal{B}_j^r,$$

where $\mathcal{B}_j^r$ is the publisher j's total investment in the routing services (i.e. bandwidth allocation on the surrogates) and $\alpha \to 0$ is present to make (P_j'') similar to the dis-

tribution sub-problem and it allows us to use the already proven results. Notice that as $\alpha \to 0$, the solution of (P_j'') will not be affected by this factor. (P_j'') minimizes the total average delay for each publisher j, given that the total cost of service does not exceed the publisher's willingness to pay. (P') and (P_j'') are analogous problems. In the first case, the publisher (or, on behalf, of the publisher the RRS) minimizes the cost subject to the condition that the average delay does not exceed the delay bound, while in the second case, the publisher minimizes its delay bound subject to the cost of service does not exceed its investment. Furthermore, in the first case, the publisher determines its delay bound so that its net benefit is maximized, and in the second case, the publisher selects its investment amount with the same objective in mind.

We have shown in the previous chapter that the game defined by (P'') and the surrogate revenue maximization problem has an equilibrium due to Theorem 1. The optimal surrogate pricing strategy is the one that completely allocates the available bandwidth capacity among the users.

The main reason (P) does not lead to an equilibrium, but (P_j'') does, is that in (P_j'') the surrogates have the information about the publishers' willingness to pay. Thus, they cannot increase the prices unboundedly. By stating their investment amount, the publishers' are essentially participating in an auction. The surrogates set prices according to the outcome of this auction. Without the publisher investment amounts or as in an auction *bids*, the surrogates are unaware how willing the publishers are for achieving their requested average delay bound. Without such knowledge it is optimal for surrogates to charge arbitrarily large prices knowing that publishers' must purchase services to satisfy their delay requirements. Then, with such high prices the publishers cannot afford to get service from the surrogates and the demand goes to zero. When the demand goes to zero, the surrogates reduce their prices to arbitrarily small values

to compete with the origin server, and this process repeats itself. Obviously, such a system does not lead to an equilibrium.

However, the results that we derived in the previous sections are still relevant, since as we will discuss in the coming section, it is usually the case that there is a general equilibrium. At the general equilibrium the participants individually have negligible effect on the outcome of the system, since there are many publishers and surrogates. When there is a general equilibrium, auctions are not needed to allocate the resources. The CDN has the sufficient knowledge of the network state to offer a *tariff* to the publishers with respect to several parameters including the requested delay bound, arrival rates, etc. Thus, the publishers can select a maximum delay bound maximizing their benefit, and the CDN can route the user requests accordingly with the objective of minimizing the total cost.

4.5 Combined Dissemination-Routing Game

In this section, we discuss the existence of the equilibrium for the combined dissemination-routing game. First, we consider the case where each agent (publisher or surrogate) can affect the outcome of the game. This corresponds to the partial equilibrium in microeconomics. Later, we discuss the existence of the general equilibrium, when there are many agents with infinitesimal effects.

4.5.1 Partial Equilibrium

In this chapter we showed that the surrogate-routing game has an equilibrium. We also showed in the previous chapter that the publisher-surrogate dissemination game has an equilibrium. These two games are inter-related. The optimal cache allocations depend on the arrival rates to the surrogates. Meanwhile, the optimal arrival rates

to the surrogates depend on the delay associated with each surrogate, which in turn depends on the cache hit probability and thus the cache allocation. We now determine some equilibrium results for this combined game.

Notice that the combined caching and routing multi-criteria optimization problem is given as,

$$\max_{\{\underline{x}_j, \underline{\lambda}_j\}} \quad U_j(x_j, \lambda_j) \tag{4.7}$$

$$\sum_j x_j^s \leq C_s^{cache}, \qquad \forall s$$

$$\sum_j \lambda_j^s \leq C_s^{BW}, \qquad \forall s,$$

where x_j^s and λ_j^s are the caching and transmission resources allocated to publisher j at the surrogate s respectively and C_s^{cache} and C_s^{BW} are the caching and bandwidth capacities of surrogate s. This optimization problem is equivalent to

$$\max_{\{\underline{x}_j\}_j} \left\{ \max_{\{\underline{\lambda}_j\}_j} \left[U_j(\underline{x}_j, \underline{\lambda}_j) \mid \sum_j \lambda_j^s \leq C_s^{BW} \right] \mid \sum_j x_j^s \leq C_s^{cache} \right\}. \tag{4.8}$$

In the non-cooperative game each server solves the following optimization problem,

$$\max_{\mathcal{B}_j^{cache}, \mathcal{B}_j^{BW}} \left\{ \max_{\underline{x}_j} \left\{ \max_{\underline{\lambda}_j} \left[U_j(\underline{x}_j, \underline{\lambda}_j) - \sum_s \lambda_j^s p_s^{BW} \mid \sum_s \lambda_j^s p_s^{BW} \leq \mathcal{B}_j^{BW} \right] \right.\right.$$

$$\left.\left. - \sum_s \lambda_j^s p_s^{cache} \mid \sum_s \lambda_j^s p_s^{cache} \leq \mathcal{B}_j^{cache} \right\} \right\}. \tag{4.9}$$

Notice that apart from the selection of the investment amounts in the caching and transmission resources, $\mathcal{B}_j^{cache}$ and $\mathcal{B}_j^{BW}$ respectively, the optimization problem in Eq. (4.9) is the dual optimization problem of Eq. (4.8), where p_s^{cache} and p_s^{BW} can

be considered as the Lagrangian constants (or shadow prices) corresponding to the constraints in Eq. (4.7). Thus, similar to the discussion in Chapter 3 if the investments are sufficiently large and if the corresponding game has an unique equilibrium, then the game theoretical solution is equal to the system optimum.

We now show that this combined game has an equilibrium for a given set of investments by showing that the corresponding best reply function is continuous. Notice that the output of the routing game is β_j^s while the output of the distribution game is x_j^s. From the previous chapter we know the best cache dissemination strategy given the gain factor β_j^s and the set of prices $\underline{p}^{cache}$ is,

$$x_j^s = \frac{\left(\frac{\beta_j^s}{p_s^{cache}}\right)^{1/\alpha_j} \mathcal{B}_j^{cache}}{\sum_{k=1}^{S} p_k^{cache} \left(\frac{\beta_j^k}{p_k^{cache}}\right)^{1/\alpha_j}},$$

where $\beta_j^s = \lambda_j^s w_j(d_{js}^{upper})/\chi_j^{1-\alpha_j}$, $\mathcal{B}_j^{cache}$ is the total investment of the publisher j in caching and d_{js}^{upper} is the average delay observed between the publisher and the surrogate (notice that links among the publishers and the surrogates form the *upper* layer in this hierarchy.) For our purposes $w_j(\cdot)$ is a concave function referring to the benefit received from reducing the average delay by the corresponding amount. For simplicity, we assume for the following discussion that $w_j(d_{js}^{upper}) = \kappa_j d_{js}^{upper}$.

The routing sub-problem solves the optimization problem (P_j'') as discussed in the previous sections. The solution for (P_j'') is

$$\lambda_j^n(s) = \frac{\left(\frac{d_{sn}^{tot}(j)}{p_s^{BW}}\right)^{1/\alpha} \mathcal{B}_j^{BW}}{\sum_k p_k^{BW} \left(\frac{d_{kn}^{tot}(j)}{p_k^{BW}}\right)^{1/\alpha}}. \tag{4.10}$$

The delay $d_{sn}^{tot}(j)$ is the total average delay observed by the users from network n and

routed to the surrogate s when they are interested in the objects from the publisher j, i.e.

$$d_{sn}^{tot}(j) = d_{js}^{upper}(1 - \Pr(hit)) + d_{sn}^{lower}. \tag{4.11}$$

Let d_{sn}^{lower} correspond to the sum of transmission and propagation delays associated with the user request originating in network n and routed to the surrogate s (notice that the user networks and the surrogates form the *lower* layer in this hierarchy.) Notice that although the transmission delay is a function of the load of the surrogate, due to the optimality of operating at the *full load* we may assume that the transmission delay equals to the full load transmission delay and is a constant. When an object from the publisher j is requested the hit probability at surrogate s is Pr(hit)= $\left(\frac{x_j^s}{\chi_j}\right)^{1-\alpha_j}$.

Thus, given x_j^s the routing sub-problem determines a new gain factor

$$\beta_j^{s\prime} = \sum_n \lambda_j^n(s) d_{js}^{upper} / \chi_j^{1-\alpha_j} \tag{4.12}$$

to be used by the distribution sub-problem. In fact, these two sub-problems can be considered as a single onto function, since given a $\beta \in B$ a new $\beta' \in B$ is determined. If this function is continuous, then from the Brouwer's Theorem it has a fixed point. Such a fixed point constitutes the equilibrium point of the combined game. In fact it is easy to see that $\beta_j^{s\prime}$ as given by Eqs.(4.10) and (4.12) is continuous in $d_{sn}^{tot}(j)$ for all set of prices $\underline{p}^{BW}$ and $d_{sn}^{tot}(j)$ is continuous for in β_j^s for all $\underline{p}^{cache}$. Thus $\beta_j^{s\prime}$ is continuous in β_j^s and a fixed point exists. We thus proved the following theorem.

Theorem 7 *The combined distribution and routing games have an equilibrium for a given set of publisher investments* $\underline{\mathcal{B}}^{cache}$ *and* $\underline{\mathcal{B}}^{BW}$.

We have performed numerical experiments to investigate the performance of this combined game theoretical algorithm as compared to the current caching systems.

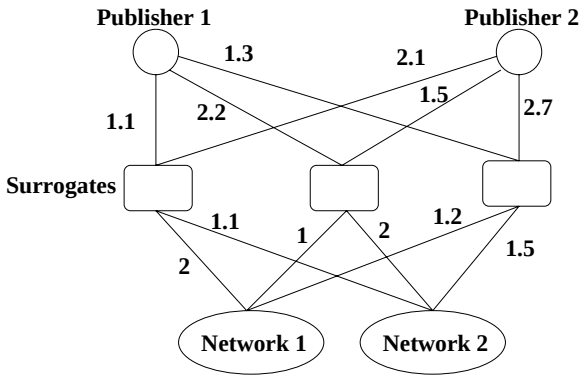

Figure 4.5: Example network for investigating the performance of the game theoretical allocation strategy.

We considered a simple network architecture, where there are two origin web servers and a CDN with three surrogates. There are two local area networks from where the user requests are generated. The duration of simulation even for such a simple network architecture is considerable, so we did not pursue more complicated network architectures. However, this network still gives a good idea about the improvements in the user delays by using game theoretical allocation strategies. The network delays among the publishers and the surrogates as well as the surrogates and the user networks are given in the Figure 4.5.

The game theoretical algorithm works as described in this section and Section 3.7. We model the current caching systems for our purposes as follows. The user requests are intercepted by the closest proxy (surrogate) to the user network. The proxies allocate caching space to the server contents with respect to the popularity of the objects and the server investments. Specifically, we assume that each proxy cache performs Least Frequently Used (LFU) cache management algorithm, which is weighted by the caching investments of the origin servers. If the requested object is found in the proxy and if the proxy is not overloaded (i.e. number of user requests are

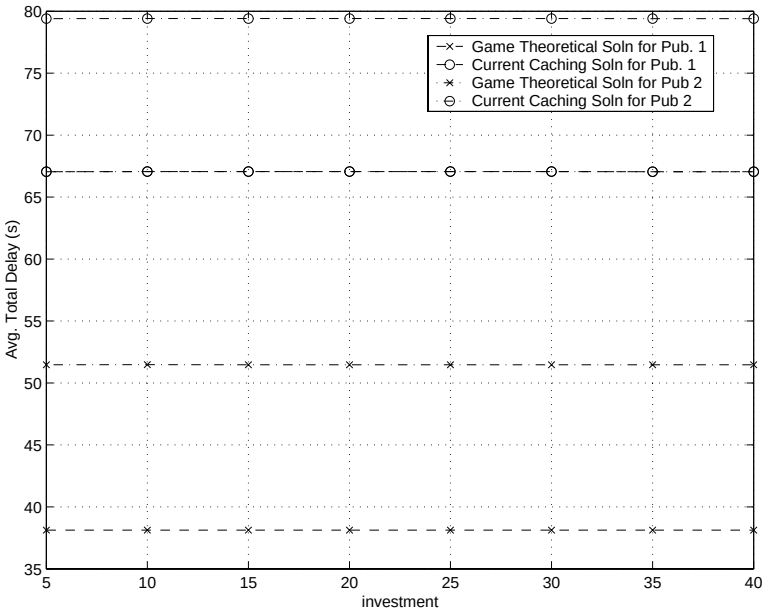

Figure 4.6: Comparison of average delays that user requests observe when they are served by a caching system. Publisher investments are the same.

not more than the available transmission/processing capacity of the proxy), then the request is immediately served. Otherwise, the user request is forwarded to the origin server. Total number of objects available in each origin server is 100, while only 10 objects can be stored in each surrogate. Each surrogate can serve at most 10 user requests at the same time. We assume that each network on average generates much more than 10 user requests at a time.

Following results depict the performance gains associated with the game theoretical algorithms for different scenarios. From Figures 4.6 and 4.7 we observe the average delay experienced by the user requests that are served by the caching infrastructure. In these examples we assume that the investment amounts of the publishers are equal. We observe the average latency experienced by the users for each publisher for varying investments. We noticed that as the investment amounts change, since the caching

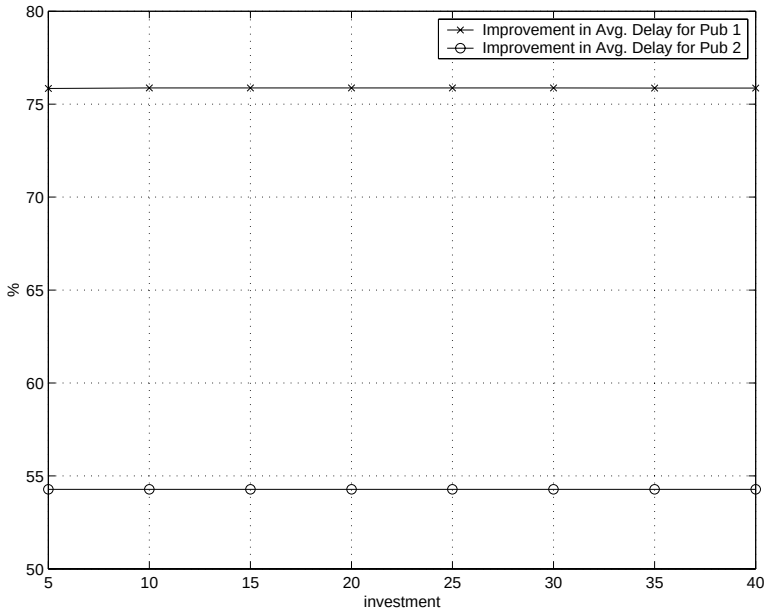

Figure 4.7: Improvement in the average latency that user requests observe when a game theoretical cache allocation strategy is used. Publisher investments are the same.

space is kept constant the total improvement remains the same.

There is considerable improvement by game theoretical CDN architecture as compared to the transparent proxy based caching system. This difference stems from several factors. First, in the current caching systems, the user request is intercepted by the proxy. This proxy may not always be the best proxy to serve this request because of its current load. Thus, if the proxy is overloaded many of the user requests has to be diverted to the origin servers increasing the latency. Second, the current caching systems allocate caching space according to such simple popularity based schemes as LRU or LFU. These algorithms do not consider the network distances of the users and the origin servers. Although there are other cache management schemes which attempt to alleviate this problem, still the proposed schemes are heuristics which are usually far from optimal.

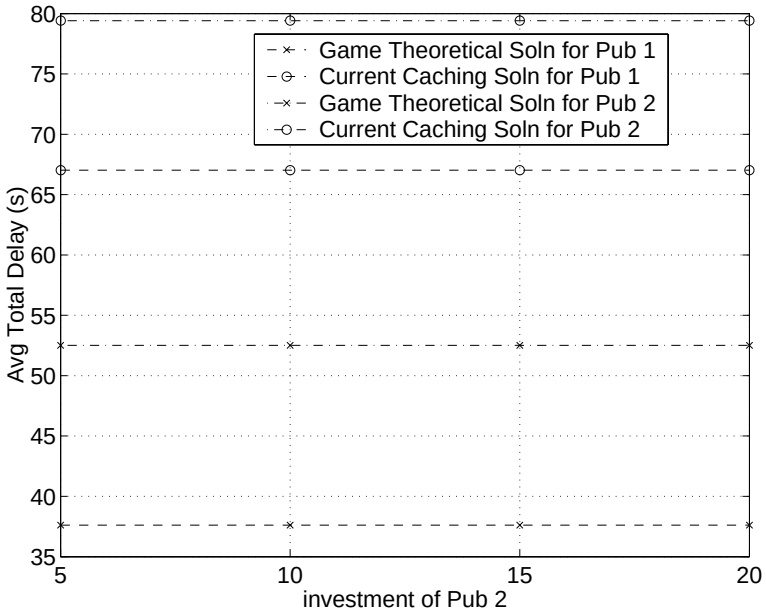

Figure 4.8: Comparison of average delays that user requests observe when they are served by a caching system. Publisher 1's investment is twice the investment of publisher 2.

Figures 4.8 and 4.9 depict the performance improvements, when publisher 1 always invests twice the amount that publisher 2 invests. Again, since the caching space is limited, we observe that the improvement among the two schemes does not change as the investments increase.

In Figure 4.10, we investigate the effect of the varying sizes of caching capacities on the average delay on both the game theoretical solution and the current caching solution. In the scenario as depicted in Figure 4.10 we assume that the size of the cache of the second surrogate is half that of the sizes of the other two surrogates. We observe that, as expected, the average delay decreases as the size of the cache increases, but the improvement of game theoretical solution remains approximately constant.

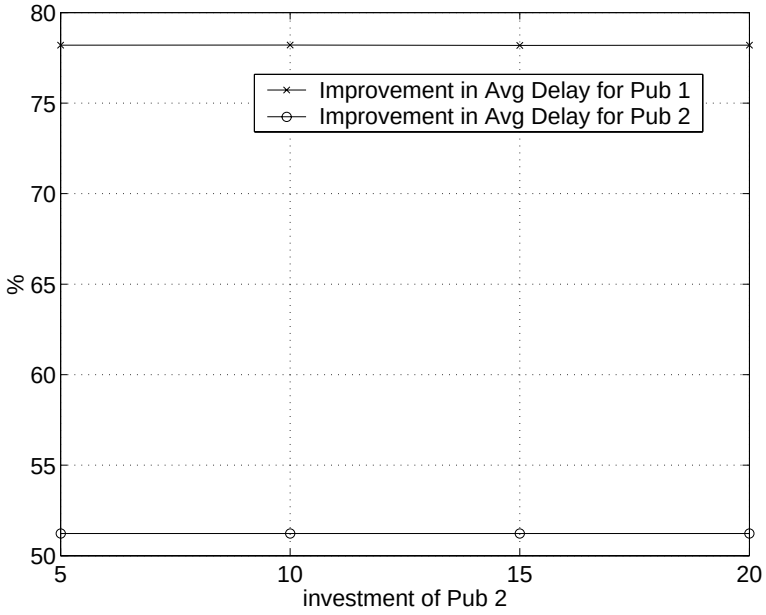

Figure 4.9: Improvement in the average latency that user requests observe when a game theoretical cache allocation strategy is used. Publisher 1's investment is twice the investment of publisher 2.

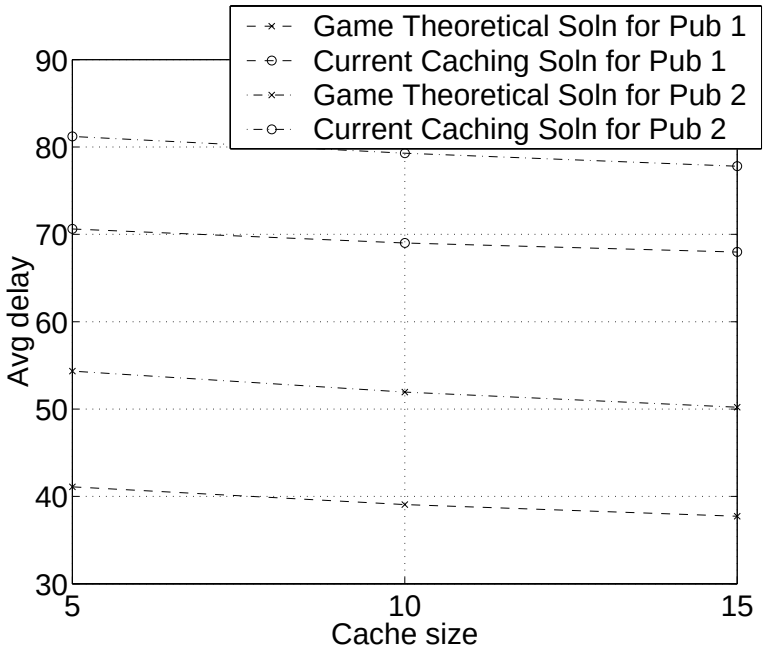

Figure 4.10: Comparison of average delays as the caching spaces of the surrogates are varied.

4.5.2 General Equilibrium

In the general equilibrium theory in economics, it is usually assumed that each individual agent in the system has infinitesimal effect on the overall outcome of the system. We may use a similar assumption when considering the content delivery problem, since in the Internet there are many publishers and surrogates and an individual publisher or surrogate cannot usually change the current network state.

Let $z_s^{cache}(\mathbf{p}^{cache}) = \sum_j x_j^s - C_s^{cache}$ be the excess demand function for the surrogate s's caching resources and let $z_s^{BW}(\mathbf{p}^{BW}) = \sum_j \lambda_j^s - C_s^{BW}$ be the excess demand function for the surrogate s's bandwidth resources.

Theorem 8 Existence of Walrasian Equilibrium [41]

A system is said to be in Walrasian equilibrium, if there exists a set of prices $\mathbf{p}^{cache}$

and $\mathbf{p}^{BW}$ *such that* $z_s^{cache}(\mathbf{p}^{cache}) \leq 0$ *and* $z_s^{BW}(\mathbf{p}^{BW}) \leq 0$ *for all* s. *A Walrasian equilibrium exists if* $\mathbf{z}^{cache}(\ p^{cache})$ *and* $\mathbf{z}^{BW}(\mathbf{p}^{BW})$ *are continuous and satisfies* $\mathbf{p}^{cache} \cdot \mathbf{z}^{cache}(\mathbf{p}^{cache}) = 0$ *and* $\mathbf{p}^{BW} \cdot \mathbf{z}^{BW}(\mathbf{p}^{BW}) = 0$.

The existence condition is satisfied, when the utility functions of the publishers are strictly increasing. Thus, by definition the combined dissemination-routing game has a Walrasian general equilibrium.

Then, given an equilibrium state, which is determined from the previous discussions, the CDN can determine a static pricing schedule and announce it to the publishers. The publishers can choose their level of service quality according to their utility and the pricing schedule. The pricing schedule consists of two parts: the dissemination cost and the bandwidth cost. The pricing function should depend on at least two parameters: the total arrival rate and the delay bound.

The requests incoming from different networks are usually non-uniform. We investigated the effect of this non-uniformity in user request distribution on the total cost. In the simulation leading to Figure 4.11, we assume that the current system load is the same as the load used in the last simulation leading to Figure 4.4. The request arrival rates for a publisher is given by the Zipf distribution with parameter β. That is, the total arrival rate to the publisher is coming from a user network with probability given by the Zipf probability considering the index of that network. As β increases the non-uniformity of the arrival rates increases as well. From Figure 4.11, we see that the total cost of servicing this publisher is relatively constant. Even if a user network contributes higher than the other user networks in the total incoming requests, the cost of service is comparable to the case when all user networks contribute equally in the incoming requests. We believe that $\beta = 1$ corresponds to sufficient skewness as observed in the Internet.

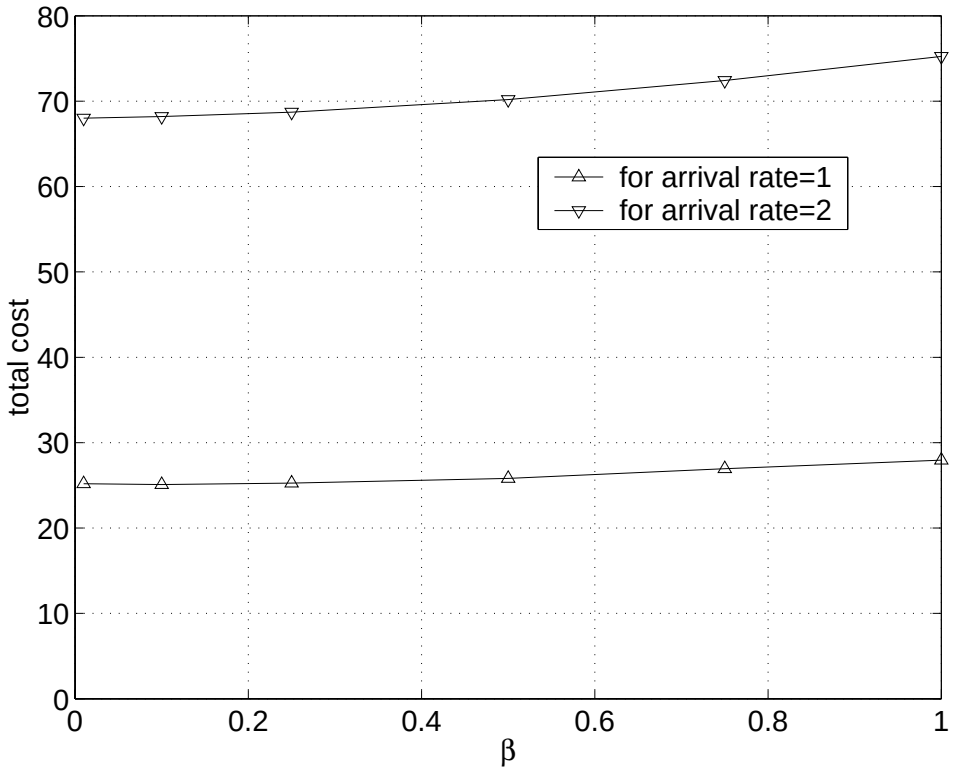

Figure 4.11: The effect of non-uniformity of user distribution in total cost.

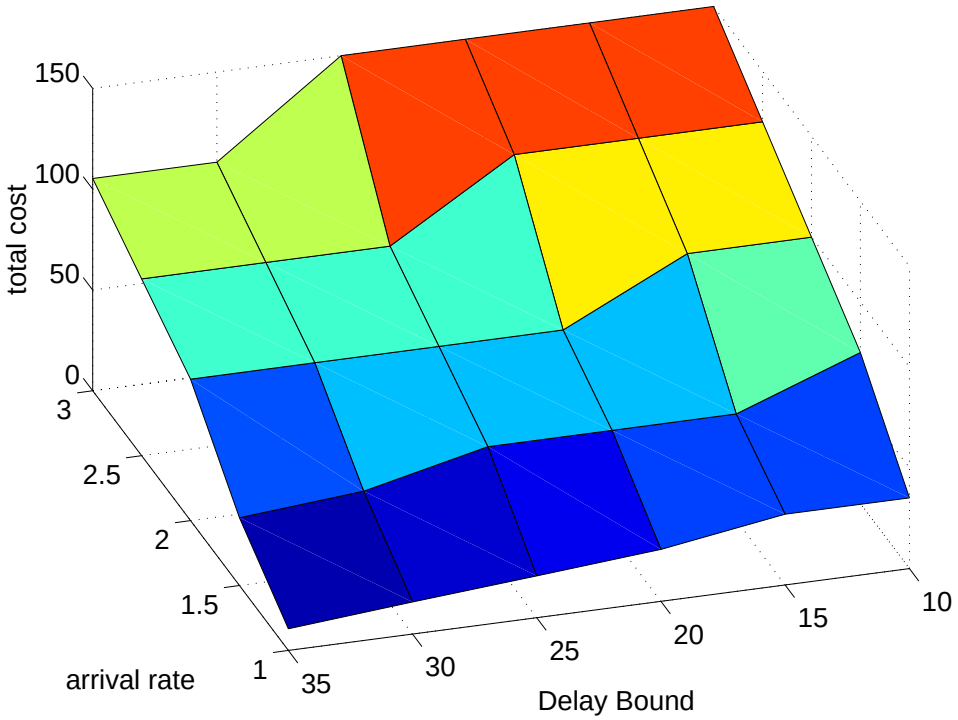

Figure 4.12: The pricing schedule.

From this observation, we can design the CDN *tariff* with respect to the total arrival rate expected and the delay bound required by the publisher. A sample *tariff* is depicted in Figure 4.12. This tariff is determined according to the total cost of service as given by the partial equilibrium reached by the network, which includes 100 publishers accessing the CDN with 10 surrogates serving 10 user networks. The request arrival patterns from the user networks are Zipf distributed with $\beta = 0.5$.

4.6 Conclusions and Future Work

In this chapter, we considered the routing of the user requests to the most appropriate surrogates. We developed an algorithm that minimizes the total cost of service, while satisfying the SLA agreed upon with the publishers. We see that with this formulation,

the system may not achieve an equilibrium. We further showed that when the surrogate charge the users with the objective of maximizing their revenue, the resulting game has an equilibrium. We also showed that the overall combined content dissemination/routing game also has an equilibrium. We analyzed the performance of this game and observed that the improvements suggested by this method are significant.

Chapter 5

Optimal Nonlinear Pricing for Multi-class Surrogates

In the previous chapters, we discussed the allocation of the caching and the bandwidth resources to the publishers with the objective of maximizing the publishers' net utilities. We used market-based methods, where the surrogates charge a fee for unit resource allocated to a publisher. The publishers can acquire as much resource as needed to maximize their own benefit in the current market state. The surrogates update their prices according to the prices of their competitors. Although we have seen that such a system can lead to an efficient resource allocation, the implementation complexity may be quite high considering the vast size of the Internet and its large user population. In this chapter, we pursue the analysis of a system, where the publishers cannot acquire the *exact* amount of resources that they would like. The surrogate offers only a limited number of service classes and the publisher can only select its service class. Within each service class, the amount of resource allocated to a publisher varies with the number of publishers subscribed to that class and the total size of the resource.

We can think of the model used in the previous sections as the *Integrated Services*

(IntServ) and the model used in this section as the *Differentiated Services (diffserv)* architectures. IntServ and diffserv architectures are developed in the IETF community for the provision of QoS in the networks. These architectures are developed for using priorities in forwarding of the packets through the network to ensure the user connections conform to specific QoS requirements. IntServ offers more strict service guarantees by allocating specific bandwidth, buffer, etc. resources at each node. Such an implementation involves high processing and implementation costs. Meanwhile diffserv offers less strict statistical service guarantees, where the user's service may fall below certain quality level with some probability. Current research efforts in network QoS has identified the provision of statistical service guarantees as a more practical approach. diffserv is an architecture developed from this observation. The *diffserv* architecture offers limited service guarantees in the form of service classes [12]. The Olympic service model was proposed as an example of diffserv architecture [39], [5]. In this model bronze, silver and gold service classes are offered to the users. The packets assigned to the gold service class usually experience lighter load than the packets assigned to the silver class. However, there are no guarantees for the service quality. Thus, the load for each service class may be high at certain times resulting in an unacceptable performance for an application even with the gold class and at other times the load may be so low that the same application can satisfactorily run over bronze class.

In the following, we will explore the same kind of paradigm for the CDNs. We assume that each surrogate offers a limited number of service classes to the publishers. The publishers have different smooth and concave willingness-to-pay functions, which is known a priori by the CDN. For simplicity the CDN assigns each publisher into one of the few available classes. The surrogates reserve fixed amount of resources for each

service class. The publishers that subscribe to the same class share the same resource. Thus, each publisher is assigned a caching space depending on the service class and the number of publishers subscribed to that class.

This architecture can be easily implemented with the current Internet state-of-the art. The so-called transparent web caching architecture can be used for this purpose. A transparent caching system acts as a router, which forwards to the Internet everything but TCP traffic for HTTP requests (requests with TCP port number 80). The TCP traffic for HTTP requests are forwarded to one or more caches. Usually, multiple caching servers are connected to the router for scalability and robustness. Thus, we may consider the surrogate as a collection of caches. A Layer-4 (L4) switch (or a router) is placed in front of these caches for routing of user requests to the appropriate cache. A Layer 4 switch routes packets according to the information available in the TCP header (and TCP is a transport layer (layer 4) protocol in ISO OSI 7 layer reference model). Each origin web site is associated with a service class by a priori subscription agreements. Meanwhile, several caches in a surrogate are assigned to serve the user requests for the web sites subscribed to a service class. The router builds two tables for request routing purposes. In the first table, the router keeps the addresses of the web sites and their associated service classes and in the second table the association of the caches with these service classes. Upon arrival of a user HTTP request, the transparent proxy checks the TCP header for the destination address (the identification for the web site). Then, it determines the service class associated with the origin server and forwards the user request to one of the caches serving that service class.

The idea of allocating fixed resources to each service class was previously discussed in the literature. In RFC 2597, the authors proposed the Assured Forwarding frame-

work for diffserv, where at each diffserv node a certain amount of forwarding resources is allocated to every class [39]. However, it is argued in [5] that this method (in the paper the so called Class-Based Allocation, CBA) leads to desired level of differentiation when the load is uniform among the classes. When the load in the higher priority classes is larger, this method cannot differentiate the traffic to allow the higher priority class to receive better service. We also acknowledge that this type of architecture cannot guarantee a service quality for a service class, since the QoS depends on the arriving load in that class. However, note that the demand for caching resources is not as bursty as the traffic in the Internet. Thus, this approach should be acceptable when the load is relatively stable in the short and long term. In the Internet there are many web sites which receive a stable stream of user traffic (e.g. CNN, Yahoo, etc.). Our method can be applied to these web sites for their day-to-day traffic. Rarely, the web sites receive *flash crowds*, when the users populate the web site to receive a particular information. For such cases, we direct the readers to the methods suggested by [46].

In the following, we first describe the system model in detail. Then, we describe the optimal pricing strategy maximizing the revenue of the CDN. We further investigate the issues in optimal pricing strategy under competition. Finally we determine a rule for specifying how many service classes to offer.

5.1 System Model

Assume that each server has an utility function (or equivalently willingness-to-pay function) $P(x)$ which represents quantitatively the benefit received by the server when x units of cache space is allocated to the server. It has been shown that users are most interested in a small portion of the content in the web server [10], [13], [51]. These studies have shown that the web user traffic can be realistically modeled by

Zipf distribution [88]. From this understanding, we may model the server's utility for the caching space as a concave monotonically increasing function. Intuitively, the server's utility increases rapidly for the low values of caching space, while the rate of increase in utility is lower when the portion of the caching space allocated to the server is high. There are infinitely many server utility functions depending on the server content, popularity, overall network state and server financial state. In order to reduce the complexity of estimating individual server utility functions, the surrogate maps all of the servers into finite number of classes. In general we may assume that there are K classes of servers with corresponding utility functions P_k, $k = 1, 2, \ldots, K$ that constitute the population of our economy. The mapping and distinction between classes can be done based on usage profiles, quality requirements, popularity, etc. The selection of optimal number of classes is discussed in section 5.4.

The surrogate has its own pricing policy based on which a user is charged a price $C(x)$ per time unit for an amount of cache space x. A server determines what kind of QoS (x) it would like based on its own utility function and the pricing policy without violating the condition $P(x) \geq C(x)$. That is, the server never pays more than the benefit it receives from a certain amount of cache space. It is usually the case that users do not have exact knowledge about their utility for a specific QoS. Rather they have a budget (or willingness-to-pay) for the service (or product). As long as they do not exceed their budget, users are willing to receive the highest QoS that is offered. Thus, a reasonable model is that the user requests the maximum cache space that it can pay for.

The surrogate assigns a certain portion of the total cache to each service class. Let b_k be the cache space assigned to the k th class of servers. Note that $0 \leq b_k \leq \mathcal{B}$, where $\mathcal{B}$ is the total surrogate storage capacity. Each server from the same class receives equal

caching space. The surrogate has prior knowledge on the server willingness-to-pay functions and the server access statistics. Let N_k be the random variable representing the total number of kth class servers accessing the surrogate at an arbitrary instant. N_k varies in time. When a new web server subscribes to the CDN's services, the address of this web site as well as the desired service class is disseminated to the surrogates. The surrogates partition their caching space among the publishers' that are in the same service class. In each partition, LFU, LRU or any other desired cache replacement policy is used to determine which objects are stored. Thus, b_k/N_k is the random variable representing the amount of cache space available to a kth class server. The objective of the surrogate is to maximize its expected revenue by selecting an appropriate *pricing* policy. In order to ensure fairness among different classes of servers, we assume that the surrogate charges all users the same amount as long as their share of cache space is the same. In this chapter, we restrict ourselves with static pricing policies, where a pricing policy is said to be static, if it does not change with respect to time or network state. The static pricing policies are usually preferred by the users due to the predictability of the costs.

5.2 Optimal Strategy for Monopolistic Surrogate

The stochastic optimization problem that surrogate solves is given as follows:

$$\text{(P)} \quad \max_{C(\cdot),\{b_k\}_{k=1}^{K}} \quad \sum_{k=1}^{K} E_{N_k}\left\{N_k C\left(\frac{b_k}{N_k}\right)\right\} \tag{5.1}$$

$$\text{subject to} \quad (1) \quad C\left(\frac{b_k}{N_k}\right) \leq P\left(\frac{b_k}{N_k}\right) \quad k = 1, \ldots, K$$

$$(2) \quad \sum_{k=1}^{K} b_k \leq \mathcal{B}.$$

The objective function is the total average revenue generated from all classes of servers given the server access statistics N_k. The surrogate server determines the pricing policy $C(\cdot)$ and the partition of the overall surrogate capacity among the classes of servers, $\{b_k\}_{k=1}^{K}$. The first constraint guarantees that the cache space allocated to a server never costs more than the utility it provides to the server. The second constraint guarantees that the total cache space allocated to each class does not exceed the available surrogate server capacity $\mathcal{B}$.

5.2.1 Properties of Optimal Pricing Strategy

If random variable N_k takes values in the range $[n_k^L, n_K^H]$, the first constraint in (P) suggests that $C(x_k) \leq P_k(x_k)$ for $\frac{b_k}{n_k^H} \leq x_k \leq \frac{b_k}{n_k^L}$. For easy demonstration purposes let $n_k^H \to \infty$. Then, $C(x_k) \leq P_k(x_k)$ for $0 \leq x_k \leq \frac{b_k}{n_k^L}$.

Notice that $\frac{b_k}{n_k^L}$ is an *upper bound* on x_k, until when $C(x_k)$ should be less than or equal to $P_k(x_k)$. Let $\mathcal{U} = \left\{ \frac{b_1}{n_1^L}, \frac{b_2}{n_2^L}, \ldots, \frac{b_K}{n_K^L} \right\}$ be the set of these upper bounds, and $u^{(m)}$ be the index of the mth smallest element in the set $\mathcal{U}$. That is, $\ldots \leq \frac{b_{u(m-1)}}{n_{u(m-1)}^L} \leq \frac{b_{u(m)}}{n_{u(m)}^L} \leq \frac{b_{u(m+1)}}{n_{u(m+1)}^L} \leq \ldots$.

Theorem 9 *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$ be the collection of utility functions of all classes. Then, for a given cache space partition $\{b_k\}_{k=1}^{K}$, the pricing policy $C(x)$ that maximizes the objective function in (P) is*

$$C(x) = \begin{cases} \min\{P_k(x) : P_k \in \mathcal{P}\}, \\ \quad if\ 0 < x \leq \frac{b_{u^{(1)}}}{n^L_{u^{(1)}}} \\ \min\{P_k(x) : P_k \in \mathcal{P} - \{P_{u^{(1)}}\}\}, \\ \quad if\ \frac{b_{u^{(1)}}}{n^L_{u^{(1)}}} < x \leq \frac{b_{u^{(2)}}}{n^L_{u^{(2)}}} \\ \vdots \\ \min\{P_k(x) : P_k \in \mathcal{P} - \{P_{u^{(1)}}, \ldots, P_{u^{(j-1)}}\}\}, \\ \quad if\ \frac{b_{u^{(j-1)}}}{n^L_{u^{(j-1)}}} < x \leq \frac{b_{u^{(j)}}}{n^L_{u^{(j)}}} \\ \vdots \\ \min\{P_k(x) : P_k \in \mathcal{P} - \{P_{u^{(1)}}, \ldots, P_{u^{(K-1)}}\}\}, \\ \quad if\ \frac{b_{u^{(K-1)}}}{n^L_{u^{(K-1)}}} < x \leq \frac{b_{u^{(K)}}}{n^L_{u^{(K)}}} \end{cases} \tag{5.2}$$

Proof From the first constraint in (P),

$$C(x) \leq \min\{P_k(x) : P_k \in \mathcal{P} - \{P_{u^{(1)}}, \ldots, P_{u^{(j-1)}}\}\}, \tag{5.3}$$

for $\frac{b_{u^{(j-1)}}}{n^L_{u^{(j-1)}}} < x \leq \frac{b_{u^{(j)}}}{n^L_{u^{(j)}}}$ and for all $k = 1, \ldots, K$. Then, $C(x)$ that maximizes the objective function in (P) is the one that satisfies the above inequality with an equality.

■

Theorem 9 states that the optimal pricing policy is discontinuous with jumps from one utility function to another and is monotonically increasing. The pricing function $C(x)$ has the structure as depicted in Figure 5.1. Theorem 9 suggests that the problem in hand can be reduced to calculate the optimal sizes of the resource partitions so that (P) is maximized. This simplified problem can further be solved if we assume that the N_k $k = 1, \ldots, K$ are independent identically distributed and the publisher utility

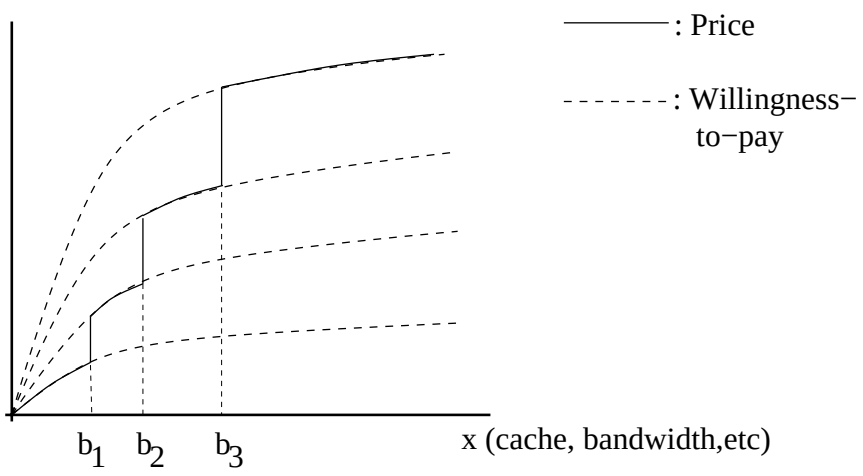

Figure 5.1: Sample pricing function.

functions are *non-crossing*, i.e. $P_{k+1}(x) > P_k(x), \forall x$ and $k = 1, \ldots, K$. In economics literature the *non-crossing* assumption for consumer demand functions is considered as the conventional approach in calculating the nonlinear tariffs [67]. In the cache allocation framework this assumption may impose some restrictions on the publisher request distributions. However, if the publisher utilities are the weighted versions of their hit probabilities, then this assumption is naturally satisfied. In many cases, the utility of a publisher is related to the cache hit probability, since for large cache hit probabilities the number of user requests served at the cache increases, and thus the user requests do not have to be forwarded to the origin server. This usually tranlates into lower average delays experienced by the users. Notice that the cache hit probability for a publisher with Zipf distribution parameter γ_i and M total objects is given as $\int_0^x Pr(u)\,\mathrm{d}u = \int_0^x \frac{(1-\gamma_i)/M^{1-\gamma_i}}{u^{\gamma_i}}\,\mathrm{d}u = \left(\frac{x}{M}\right)^{1-\gamma_i}$. Thus, the cache hit probability curves of two publishers with similar number of available objects do not intersect each other as long as $\gamma_i \neq \gamma_j$, for $i \neq j$.

Theorem 10 *Let $P_{k+1}(x) > P_k(x)$, $\forall x$ and $k = 1, \ldots, K$. Assume that random vari-*

able N_k takes values in the range $[1, \infty)$ and N_k's are i.i.d. Then the optimal cache space partition of surrogate capacity among different classes of servers $\{b_k^*\}$ satisfies $b_1^* < b_2^* < \ldots < b_K^*$.

Proof Assume that $b_1 < b_2 < \ldots < b_{i-1} < b_j < b_{i+1} < \ldots < b_{j-1} < b_i < b_{j+1} < \ldots < b_K$ is an optimal cache space partition, where $j > i$. Let $C(x)$ be the optimal pricing policy (determined from the previous Theorem) corresponding to this cache space partition.

$$C(x) = \begin{cases} P_1(x) & 0 < x \leq b_1 \\ \vdots & \\ P_{i-1}(x) & b_{i-2} < x \leq b_{i-1} \\ P_i(x) & b_{i-1} < x \leq b_{j-1} \\ P_i(x) & b_{j-1} < x \leq b_i \\ P_{j+1}(x) & b_i < x \leq b_{j+1} \\ \vdots & \end{cases} . \tag{5.4}$$

Consider an alternative cache space allocation for the same set of utility functions $\{b_k^*\}$ where $b_k^* = b_k$ for $\forall k \neq j, i$. $b_i^* = b_j$, and $b_j^* = b_i$. Notice that $b_1^* < b_2^* < \ldots < b_K^*$. The optimal pricing policy with this cache space partition is

$$C^*(x) = \begin{cases} P_1(x) & 0 < x \leq b_1^* \\ \vdots & \\ P_k(x) & b_{k-1}^* < x \leq b_k^* \\ \vdots & \\ P_K(x) & b_{K-1}^* < x \leq b_K^* \end{cases} . \tag{5.5}$$

Compare the objective function in (P) for mth class of servers for both of these partitions. For $m < i$, $E\left\{N_m C\left(\frac{b_m}{N_m}\right)\right\} = E\left\{N_m C^*\left(\frac{b_m^*}{N_m}\right)\right\}$. Let $m \geq i$. Let $q_m(n)$

be the probability distribution function for N_m.

$$
\begin{aligned}
E\left\{N_m C\left(\frac{b_m}{N_m}\right)\right\} &= \sum_{n<\frac{b_m}{b_{m-1}}} nP_i\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \sum_{\frac{b_m}{b_{m-1}}\le n<\frac{b_i}{b_{i-1}}} nP_i\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \quad \cdots \\
&+ \sum_{\frac{b_i}{b_{i-1}}\le n<\frac{b_{i-1}}{b_{i-2}}} nP_{i-1}\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \quad \cdots \qquad (5.6) \\
E\left\{N_m C^*\left(\frac{b_m}{N_m}\right)\right\} &= \sum_{n<\frac{b_m}{b_{m-1}}} nP_m\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \sum_{\frac{b_m}{b_{m-1}}\le n<\frac{b_{m-1}}{b_{m-2}}} nP_{m-1}\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \quad \cdots \\
&+ \sum_{\frac{b_{i+1}}{b_i}\le n<\frac{b_i}{b_{i-1}}} nP_i\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \sum_{\frac{b_i}{b_{i-1}}\le n<\frac{b_{i-1}}{b_{i-2}}} nP_{i-1}\left(\frac{b_m}{n}\right) q_m(n) \\
&+ \quad \cdots \qquad (5.7)
\end{aligned}
$$

For $\{b_k\}$ to be optimal (as compared to $\{b_k^*\}$), $P_i(x) \ge P_l(x)$ for $m \ge l > i$, i.e. each term in equation (5.6) should be higher than each term in equation (5.7). This contradicts with the hypothesis that $P_i(x) < P_j(x)$ for $i < j$. ■

This theorem states that regardless of server access distributions, the higher paying

class of servers should receive a larger portion of the surrogate capacity. Theorems 9 and 10 together define the pricing function $C(x)$ except for $\{b_k\}_{k=1}^K$. Note that given the utility functions $P_k(x)$, $k = 1, 2 \ldots, K$ are concave and continuous, the resulting objective function in optimization problem (P) is also continuous in b_k. We can use Lagrangian methods to determine the optimal cache partition $\{b_k\}_{k=1}^K$.

We determined the solution of (P) for logarithmic utility functions where $P_k(x) = \alpha_k \log(x+1)$. We assumed, without loss of generality, that $\alpha_k < \alpha_{k+1}$. After straightforward but tedious calculations, one arrives at the following set of nonlinear equations described by (5.8) and (5.9). We assume for simplicity and compactness, the server access distributions $N_k,\ k = 1, \ldots, K$ are continuous. Let μ be the Lagrangian constant, and $f(\cdot)$ be the identical probability density function of the server access distributions $N_k,\ k = 1, \ldots, K$.

$$\begin{aligned}
&\frac{\alpha_j}{b_j}\int_0^{b_j/b_{j-1}} nf(n)\,dn + \frac{\alpha_{j-1}}{b_j}\int_{b_j/b_{j-1}}^{b_j/b_{j-2}} nf(n)\,dn \\
&\quad + \quad \cdots \\
&\quad + \quad \frac{\alpha_2}{b_j}\int_{b_j/b_2}^{b_j/b_1} nf(n)\,dn + \frac{\alpha_1}{b_j}\int_{b_j/b_1}^{\infty} nf(n)\,dn \\
&\quad + \quad \sum_{k=1}^{j-1} \frac{b_j}{b_k^2} f\left(\frac{b_j}{b_k}\right)[\alpha_{k+1} - \alpha_k]\log(b_k+1) \\
&\quad + \quad \sum_{k=j+1}^{K} \frac{b_k^2}{b_j^3} f\left(\frac{b_k}{b_j}\right)[\alpha_j - \alpha_{j+1}]\log(b_j+1) \\
&\quad = \quad \mu,\ j = 1, \ldots, K. \qquad (5.8)
\end{aligned}$$

$$\sum_{k=1}^{K} b_k = \mathcal{B}. \qquad (5.9)$$

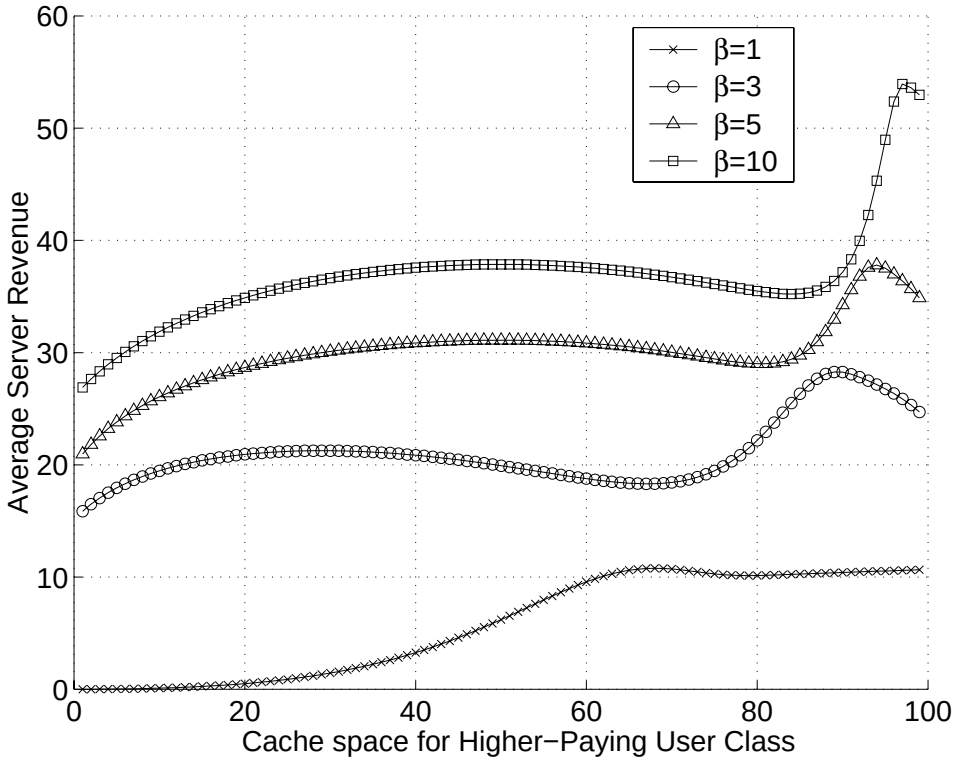

Figure 5.2: Change of average surrogate revenue with respect to surrogate partition for varying β.

5.2.2 Numerical Results

We have investigated the solution of this Lagrangian optimization problem numerically. It is reasonable to assume that the number of servers accessing the surrogate at a particular time is Poisson distributed. In our analysis, we assume that server distribution is continuous and can be represented by Rayleigh distribution, which can be considered as the continuous interpolation of Poisson distribution. We determine the optimal cache partition by simultaneously solving the set of nonlinear equations described by (5.8) and (5.9).

In Figure 5.2, we observe the effect of increasing number of servers requesting service on the optimal partition of a surrogate with 2 service classes. The server utility

functions are $P_1(x) = 1 \cdot \log(x+1)$, and $P_2(x) = 2 \cdot \log(x+1)$. β is the parameter for Rayleigh distribution that roughly corresponds to the mean number of servers requesting service at the current slot. In Figure 5.2, we notice that as β increases, the optimal partition tends to give a larger portion of the surrogate server space to the service class with higher utility. In fact, beyond some certain β, the optimal policy for a surrogate is to serve only the service class with higher willingness-to-pay. This is reasonable considering that when the average number of servers requesting service is high, the probability of receiving no server request is low. If we can assure that there is always going to be a higher paying class of servers in the system, then it is optimal to serve only higher paying class of servers. This type of behavior may result in a system where *less popular* servers are unable to get any caching service. Fortunately, in real world the number of users belonging to each service class is not identically distributed. Servers with higher willingness-to-pay or that are highly popular are always much fewer.

Table 5.1 depicts the optimal cache partition and the revenue corresponding to this partition for a monopolistic surrogate with capacity 100 units. There are 2 service classes with willingness-to-pay functions $P_1(x)$ and $P_2(x)$.

2 classes	b_1	b_2	revenue
$\beta = 1$	33.8678	66.1322	10.6477
$\beta = 3$	10.3475	89.6525	28.2833
$\beta = 5$	6.9070	93.0929	37.6394
$\beta = 10$	4.0336	95.9665	52.2785

Table 5.1: Optimal cache partition for a monopolistic surrogate when there are 2 service classes.

In Table 5.2 we investigate the optimal cache partition when there are 3 service

classes. Servers from each class request service with respect to Rayleigh distribution with parameter β. The capacity of the surrogate is still 100 units.The willingness-to-pay function for each service class is $P_1(x) = 1 \cdot \log(x+1)$, $P_2(x) = 2 \cdot \log(x+1)$ and $P_3(x) = 3 \cdot \log(x+1)$. It is interesting to note that when there are more than two service classes, the classes other than the highest paying service class receive roughly the same amount of cache space as β increases. Thus, it may be better to serve the lower paying service classes in a single service class. For this model, we may conjecture that implementing a two service class system is sufficient to achieve a near-optimal result.

3 classes	b_1	b_2	b_3	revenue
$\beta = 2$	10.2091	13.1847	76.6062	28.0497
$\beta = 3$	8.9656	9.3079	81.7265	41.6741
$\beta = 5$	6.3136	6.3557	87.3312	57.1166

Table 5.2: Optimal cache partition for a monopolistic surrogate when there are 3 service classes.

5.2.3 Discussion of the System

Even though it may occur very rarely, this scheme may result in low utilization of the surrogate capacity. Notice that our results depend on the fact that there is always at least one publisher per class. If this is not true, then the cache space allocated to a service class remains unused. This leads to under-utilization of the surrogate capacity. For a very popular surrogate server this may not cause any problem. One method to circumvent this problem is to reduce the number of different classes, and thus to reduce the probability that there is no publishers for a class. It is true that higher the

number of classes, lower the surrogate utilization is. However at the same time, lower the number of classes, lower the surrogate revenue is, since we have to represent all publishers with very few number of willingness-to-pay (utility) functions. Hence, the partitioning of the publishers into multiple surrogate classes is an additional design constraint in the maximization of surrogate revenue. We offer a possible solution to this problem in the last section. However, notice that these cases occur rarely in a caching framework, where the demand for the resources is much more stable than the actual Internet traffic as considered in the flow control problem. Furthermore, in the caching framework the contracts are usually much longer term, which ensures the stability.

5.3 Optimal Surrogate Partition Under Competition

In this section we consider the optimal pricing strategies of two competing surrogates. Let $C_1(x)$ and $C_2(x)$ denote the pricing functions of the two competing surrogates. Each surrogate offers in total $\mathcal{B}$ units of cache space to their servers. N_1^1 and N_2^1 denote the number of servers from each class accessing the surrogate 1, while N_1^2 and N_2^2 denote the number of servers from each class accessing the surrogate 2. Notice that $N_1^1 + N_1^2 = N_1$ and $N_2^1 + N_2^2 = N_2$, where N_1 and N_2 are the random variables corresponding to the number of servers accessing the surrogates at a particular time from each class. b_1^1 and b_2^1 denote the cache space reserved for each service class on the surrogate 1, and b_1^2 and b_2^2 denote the cache space reserved for each service class on the surrogate 2.

The strategy for a server is to subscribe to a surrogate and to select a class that maximizes its share of the cache space. When many servers access a particular surrogate, the surrogate will get congested and the share of cache space for each server

will become less. In that case at later time slots some of the servers will switch to the competing surrogate. An equilibrium, when the number of servers switching from one surrogate to another is the same exists, when the same class of servers receive equal share of cache space in each surrogate. That is, when $\frac{b_1^1}{N_1^1} = \frac{b_1^2}{N_1^2}$, and $\frac{b_2^1}{N_2^1} = \frac{b_2^2}{N_2^2}$. Notice that at equilibrium $N_1^2 = \frac{N_1}{1+\frac{b_1^1}{b_1^2}}$ and $N_2^2 = \frac{N_2}{1+\frac{b_2^1}{b_2^2}}$. At equilibrium the optimal pricing strategy for provider $i = 1, 2$, given the pricing strategy of the competing provider $j \neq i$ is determined by solving the following optimization problem.

$$\max_{b_1^i, b_2^i} \left\{ \frac{b_1^i}{b_1^i + b_1^j} E\left[N_1 C_i \left(\frac{b_1^i + b_1^j}{N_1} \right) \right] \right.$$

$$\left. + \quad \frac{b_2^i}{b_2^i + b_2^j} E\left[N_2 C_i \left(\frac{b_2^i + b_2^j}{N_2} \right) \right] \mid b_1^i + b_2^i = \mathcal{B} \right\}. \tag{5.10}$$

Assume that surrogate 2 is the new entrant to the market. Initially, surrogate 2 determines a pricing strategy by solving the above equation when the surrogate 1 has implemented optimal monopolistic pricing strategy. However, provider 1 will take action as provider 2 announces its pricing strategy. Both surrogates will iteratively re-calculate their strategy given the strategy of its competing counterpart. An important question is then, whether a pair of pricing strategies exits when neither of the two providers are willing to change their strategy unilaterally. In game theory such a situation is called Nash equilibrium. In the following theorem, we show that abovementioned 'pricing war' leads to a Nash equilibrium.

Theorem 11 *Let b_1^* and b_2^* be the optimal solution to*

$$\max_{b_1, b_2} \left\{ E\left[N_1 C \left(\frac{b_1}{N_1} \right) \right] + E\left[N_2 C \left(\frac{b_2}{N_2} \right) \right] \mid b_1 + b_2 = 2\mathcal{B} \right\}, \tag{5.11}$$

then $\frac{b_1^*}{2} = b_1^1 = b_1^2$ *and* $\frac{b_2^*}{2} = b_2^1 = b_2^2$ *is a cache partition that is also a Nash equilibrium, if*

$$\frac{\partial f}{\partial b}(b_1^*) + \frac{\partial f}{\partial b}(2\mathcal{B} - b_1^*) < 0, \tag{5.12}$$

where $f(b) = E\left[NC(b/N)\right]$.

Proof Assume that $\frac{b_1^*}{2} = b_1^1 = b_1^2$ and $\frac{b_2^*}{2} = b_2^1 = b_2^2$ is a surrogate partition that is not a Nash equilibrium. Then, without loss of generality surrogate 1 can increase its revenue by choosing a surrogate partition $\frac{b_1^*}{2} + \Delta$. The revenue of provider 1 with this surrogate partition is

$$\frac{b_1^*/2 + \Delta}{b_1^* + \Delta} f(b_1^* + \Delta) + \frac{\mathcal{B} - b_1^*/2 - \Delta}{2\mathcal{B} - b_1^* - \Delta} f(2\mathcal{B} - b_1^* - \Delta). \tag{5.13}$$

The net increase in the revenue of provider 1 is

$$\begin{aligned} &\frac{b_1^*/2 + \Delta}{b_1^* + \Delta} f(b_1^* + \Delta) - \frac{1}{2} f(b_1^*) \\ &+ \frac{\mathcal{B} - b_1^*/2 - \Delta}{2\mathcal{B} - b_1^* - \Delta} f(2\mathcal{B} - b_1^* - \Delta) - \frac{1}{2} f(2\mathcal{B} - b_1^*). \end{aligned} \tag{5.14}$$

Multiplying equation (5.14) by $\frac{1}{\Delta}$ and taking the limit as $\Delta \to 0$, we get

$$\frac{1}{2}\frac{\partial f}{\partial b}(b_1^*) + \frac{1}{2}\frac{\partial f}{\partial b}(2\mathcal{B} - b_1^*). \tag{5.15}$$

It is clear that if $\frac{b_1^*}{2}$ is a Nash equilibrium, then the expression in equation (5.15) is less than zero. ■

This theorem suggests that at equilibrium servers receive a share of surrogate server space as if there is a single surrogate offering a larger surrogate capacity. Servers' share of cache space increase, since a competing surrogate introduces additional surrogate

server space. The surrogates' revenues increase, since there are fewer servers subscribing to each surrogate at any particular time resulting in increased charges per server received due to higher per server cache share. Thus, the result of the competition of different surrogates is the same as peering.

We tested this result for log-utility functions through numerical analysis. There are 2 classes of servers with willingness-to-pay functions $P_1(x) = 1 \cdot \log(x)$ and $P_2(x) = 2 \cdot \log(x)$. There are two classes of service, and servers of each class request access to the surrogate with respect to Rayleigh distribution with parameter $\beta = 3$. The optimal surrogate partition, when there is a single surrogate offering in total 200 units of cache space to the servers is $b_1 = 22.43$ and $b_2 = 177.57$. We observed that in 12 iterations we approached the Nash equilibrium solution suggested by the previous theorem.

5.4 Optimal Number of Partitions

Previous sections discussed the optimal size of the partitions and the pricing strategy when the number of user classes are given. In this section, we investigate the optimal number of resource partitions maximizing the revenue. In order to maintain tractability, we assume a log-utility function $U(\theta) = \theta \log(k(q) + 1)$, where k is the dis-benefit of observing congestion q at the resource, θ is the preference of the user for the lack of congestion. q can also be interpreted as the number of users sharing the resource. Let $p \log(k+1)$ be the subscription price for the resource. In order to reflect the range of preferences in the simplest manner, we assume that there is a continuum of users whose θ parameters form a population with distribution $f(\theta)$. Notice that according to our utility function, users with inelastic preferences will have high values of θ. The users, whose θ parameters form a population distribution which is distributed on the

iteration	provider 1	provider 2
0	[10.3475 89.6525]	-
1	[10.3475 89.6525]	[12.4239 87.5761]
2	[10.7223 89.2777]	[12.4239 87.5761]
3	[10.7223 89.2777]	[12.1263 87.8737]
4	[10.9582 89.0418]	[12.1263 87.8737]
5	[10.9582 89.0418]	[11.9388 88.0612]
6	[11.1270 88.8230]	[11.9388 88.0612]
7	[11.1270 88.8230]	[11.7974 88.2026]
8	[11.2429 88.7571]	[11.7974 88.2026]
9	[11.2429 88.7571]	[11.7014 88.2986]
10	[11.3206 88.6794]	[11.7014 88.2986]
11	[11.3206 88.6794]	[11.4938 88.5062]
12	[11.4853 88.5147]	[11.4938 88.5062]

Table 5.3: The optimal cache space partition of competing surrogates. At each iteration one of the surrogate updates its cache space partition.

interval $[0, 1]$ according to distribution $f(\theta)$.

The congestion k is a function of total number of users in the system. Let, without loss of generality, $p_1 < p_2$ denote the prices of two equal size, C, partitions of the resource. Notice that users with preference $\theta < \theta_1$, where $\theta_1 \log(k(\int_{\theta_1}^{\theta_2}) + 1) - p_1 \log(k(\int_{\theta_1}^{\theta_2} f(\theta)\,\mathrm{d}\theta) + 1) = 0$ do not subscribe to the resource since the net benefit is negative. Notice that this relationship requires that $p_1 = \theta_1$. Similarly, the users with preference $\theta_1 < \theta < \theta_2$, where $\theta_2 \log(k(\int_{\theta_1}^{\theta_2} f(\theta)\,\mathrm{d}\theta) + 1) - p_1 \log(k(\int_{\theta_1}^{\theta_2} f(\theta)\,\mathrm{d}\theta) + 1) = \theta_2 \log(k(\int_{\theta_1}^{\theta_2} f(\theta)\,\mathrm{d}\theta) + 1) - p_2 \log(k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta) + 1)$, subscribe to the lower priced

partition. The profit for the partitioned resource is given as

$$\begin{aligned}\pi_2 &= \int_{\theta_1}^{\theta_2} f(\theta)\,\mathrm{d}\theta\ p_1 \log(k(\int_{\theta_1}^{\theta_2} f(\theta)\,\mathrm{d}\theta)+1)+ \\ &\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta\ p_2 \log(k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)+1).\end{aligned}$$

We search for the conditions for which the profit by offering two classes of service is higher than offering a single class of service. Consider the case when $p_1 = p_2$. Notice that this case corresponds to offering a single class of service. If we can find another solution with higher profit, then we show that it is optimal to offer multiple service classes. We show that by considering the change in profit for a given $p_1 = \theta_1 =$ *constant* is positive when p_2 is increased, i.e. $\mathrm{d}\pi > 0$, when $\mathrm{d}p_2 > 0$.

For $p_1 = p_2$,

$$\begin{aligned}\frac{\mathrm{d}\pi}{\mathrm{d}p_2} &= -f(\theta_1)\frac{\mathrm{d}\theta_1}{\mathrm{d}p_2}\theta_1 \log(k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)+1)+ \\ &(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)\theta_1 \frac{k'(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)}{k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)+1}(-f(\theta_1)\frac{\mathrm{d}\theta_1}{\mathrm{d}p_2})+ \\ &(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)\log(k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)+1),\end{aligned}$$

and assuming that θ_1 is constant

$$\mathrm{d}\pi = (\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)\log(k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)+1)\,\mathrm{d}p_2. \tag{5.16}$$

Thus, we see that for $\mathrm{d}p_2 > 0$ $\mathrm{d}\pi > 0$, if $k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta) > 0$. A good choice for the congestion function k is $k(q) = \frac{C}{q}$, where q is the number of users subscribed to the resource, and C is the limited resource capacity.

We should also note that such congestion function $k(q) = \frac{C}{q}$ may result in the provision of a single service class if the users have a more conventional utility function $U(\theta) = \theta \cdot \frac{C}{q}$ and charged a fixed price p. This result is due to the following lemma, which also identifies a case where the result of Chander and Leruth [16] does not hold.

Lemma 4 *Let $U(\theta) = \theta \cdot k(q) - p$. The congestion function $k(q)$ can be any real valued function satisfying:*

1. *k is at least twice differentiable;*
2. *$k(q) > 0$ and $dk(q)dq < 0$ for all $q > 0$;*
3. *$k(0) < \infty$.*

If $k'(q)q = -k(q)$, then it is not optimal to sell the same product with different qualities.

Proof Assume that there are two products priced at p_1 and p_2, and without loss of generality $p_1 > p_2$. The users with preference factor θ_2 are indifferent of subscribing to product 2 or not subscribing to any of the products. Notice that the users with preference $\theta > \theta_2$ do not subscribe to any of the products. That is,

$$U(\theta_2) = \theta_2 k(q_2) - p_2 = 0. \tag{5.17}$$

On the other hand users with preference θ_1 are indifferent of subscribing to either product 1 or product 2. That is,

$$U(\theta_1) = \theta_1 k(q_2) - p_2 = \theta_1 k(q_1) - p_1. \tag{5.18}$$

From equations (5.17) and (5.18) we can determine p_1 and p_2 as,

$$p_1 - p_2 = \theta_1(k(q_2) - k(q_1)), \tag{5.19}$$

$$p_2 = \theta_2 k(q_2). \tag{5.20}$$

Our objective is to determine the set of prices p_1, p_2 that maximize the total profit $\pi = p_1 q_1 + p_2 q_2$. Noticing that p_1 and p_2 are functions of the market partitions θ_1 and θ_2, we consider our optimization over θ_1 and θ_2. Let $f(\theta)$ be the number of users with θ preference. Then $q_1 = \int_{\theta_1}^{1} f(\theta)\,\mathrm{d}\theta$ and $q_2 = \int_{\theta_2}^{\theta_1} f(\theta)\,\mathrm{d}\theta$. Profit is

$$\begin{aligned} \pi &= \theta_2 k(q_2) q_2 + \theta_1 (k(q_2) - k(q_1)) q_1 + \theta_2 k(q_2) q_1 \\ &= \theta_2 k(q_2)(q_1 + q_2) + \theta_1 (k(q_2) - k(q_1)) q_1. \end{aligned} \tag{5.21}$$

Denote the differentiation of arbitrary function $g(\theta_1)$ with respect to θ_1 as $g' = \frac{\partial g(\theta_1)}{\partial \theta_1}$. Taking the derivative of π with respect to θ_1, and equating to zero, $\pi' = 0$,

$$\begin{aligned} \pi' = {} & \theta_2 k'(q_2) q_2' (q_1 + q_2) + \theta_2 k(q_2)(q_1' + q_2') \\ & + (k(q_2) - k(q_1)) q_1 + \theta_1 (k(q_2) - k(q_1)) q_1' \\ & + \theta_1 (k'(q_2) q_2' - k'(q_1) q_1') q_1 = 0. \end{aligned} \tag{5.22}$$

$$\begin{aligned} \pi' = {} & (\theta_2 (q_1 + q_2) + \theta_1 q_1) k'(q_2) q_2' - \theta_1 k'(q_1) q_1' q_1 \\ & + \theta_2 k(q_2)(q_1' + q_2') + (k(q_2) - k(q_1)) q_1 \\ & + \theta_1 (k(q_2) - k(q_1)) q_1' = 0. \end{aligned} \tag{5.23}$$

Since $k'(q) q' < 0$ from assumptions, we see that the first term in (5.23) is negative while the second one is positive. The third term is equal to zero, since $q_2' = -q_1' = f(\theta_1)$. The fourth term is negative since $p_1 > p_2$ and thus $k(q_1) > k(q_2)$. The last term in (5.23) is positive, since q_1' is negative. As it stands we may find q_1, q_2 solving (5.23) such that $q_1 \neq q_2$.

If $k'(q_1) q_1 = -k(q_1)$, then second term cancels with the last term, resulting in

$$\begin{aligned} \pi' = {} & (\theta_2 (q_1 + q_2) + \theta_1 q_1) k'(q_2) q_2' \\ & + (k(q_2) - k(q_1)) q_1 + \theta_1 k(q_2) q_1' = 0. \end{aligned} \tag{5.24}$$

All terms in (5.24) are negative, which suggests that no interior point solution exists.

■

For $k(q) = \frac{C}{q}$, we see that it is always optimal to offer two service classes, since $(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)\log(k(\int_{\theta_2}^{1} f(\theta)\,\mathrm{d}\theta)+1)\,\mathrm{d}p_2 > 0$, for all $\mathrm{d}p_2 > 0$. This result can be extended to more than two classes by using a recursive argument. For example, assume that there are 3 classes and and show that $\mathrm{d}\pi > 0$, when $p_1 = p_2 = p_3$ and θ_1 and θ_2 are kept constant.

This result is a special case as discussed by Chander and Leruth in [16]. The authors have shown in [16] that a profit maximizing monopolist will charge the maximum number of different prices, and hence offer the maximum number of sub-networks with different qualities. However, in [16] the authors used a simple pricing policy where the users of each class are charged a constant price regardless of their quality of service. In our work we considered a nonlinear pricing policy within each service class.

We have shown that given an available resource capacity, it is a revenue maximizing strategy to offer as many service classes as possible. However, as discussed previously, another consideration for a resource provider is the utilization of the resources. Note that a low resource utilization means waste of investment for the unused portion. Thus, a resource provider would like to maximize its utilization as well. In Figure 5.3 we show the change in the resource utilization for increasing number of service classes. In this experiment, we assume that user requests arrive according to Poisson distribution with rate λ. Note that in this experiment λ is the aggregate arrival rate for all types of publishers. The probability of a publisher subscribing to a service class is the same probability for all classes. The capacity of each service class is assumed to be the same. We notice that as the number of offered service classes increase, the utilization decreases exponentially. However, as the arrival rate increases the

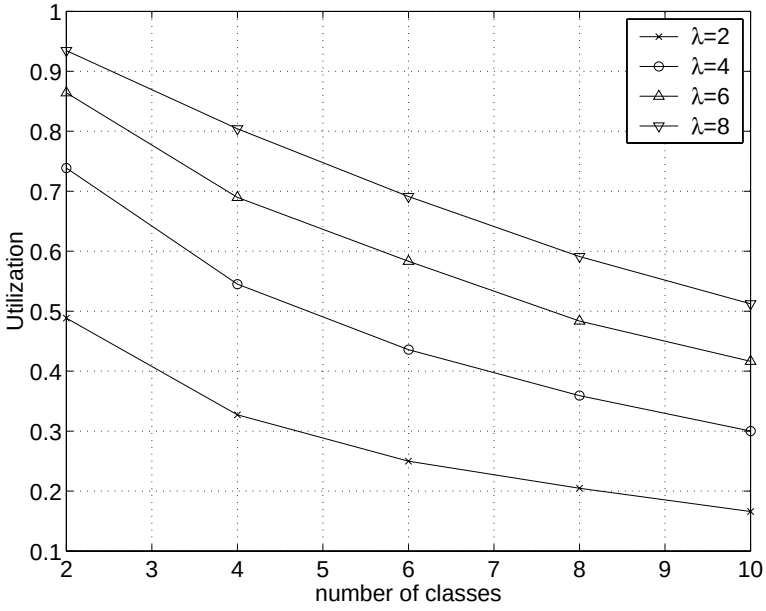

Figure 5.3: Utilization of resource for increasing number of service classes.

utilization increases as well. Thus, more number of classes can be offered efficiently when more users subscribe to the system.

A reasonable objective for the surrogate is to maximize its *profit*, which is the revenue gained by providing differentiated services less the investment required to provide the necessary resources. By multiplying the ordinate of Figure 5.3 by an appropriate weighting factor, which may depend on the unit costs of resources, the surrogate can estimate its investment. Then, by using the optimal pricing strategy discussed in the previous section, the surrogate can choose an optimal number of service classes maximizing its *profit*.

5.5 Conclusions and Future Work

In this chapter we analyzed a pricing scheme for the extension of *diffserv* architecture in the CDNs. In this architecture, the publishers (web sites) subscribe to one of the

few service classes offered by the surrogate. Within each service class, the amount of resource allocated to a publisher depends on the number of publishers subscribed to that class as well as the amount of resource the surrogate allocates to that service class. We determined the optimal nonlinear tariff for the surrogate maximizing its revenue. The optimal tariff follows the willingness-to-pay functions of the publishers, but has jumps at the points denoting the size of resource allocated to each class. At each jump the price function increases to the willingness-to-pay function of the higher service class. Thus, the problem reduces to determining the optimal partition of the resource among service classes. We also analyzed the system under competition and observed that under competition the individual resource allocations converge to the optimal allocation set by a monopoly with an aggregate resource size of two competing surrogates. Furthermore, we described a method for determining the appropriate number of service classes depending on the received revenue and network efficiency.

It should be clear to the reader that these results are applicable not only to the pricing and allocation of caching resources in surrogates but also to any other congestible resource allocation problems such as the sharing of bandwidth at the edges (ISPs) and in general for pricing of commodities whose quality varies with demand. An immediate extension of this work can be thought of by considering different user objectives such as the maximization of net benefit instead of the maximization of the share of the resource. In case the users are aware of their utility functions such an objective makes sense.

Chapter 6

Conclusions and Future Work

In this dissertation, we investigated the dynamics of resource allocation in content delivery networks. Our motivation for this study was to analyze the effects of high-level user decisions on the lower level network performance. Notice that the Internet has evolved from a research-oriented academic network into an infotainment-oriented commercial network. Academic, business and personal users form the large population of the Internet. Not all users value the Internet services the same, because of the importance of the applications that use these services. For example, a user accessing from home may use the Internet for leisure and thus is not always willing to pay for high quality Internet experience. Meanwhile, a company may use the Internet to connect its LANs at several geographically distant locations, and require high quality service for effective continuation of its business. In this framework the user performance is usually better measured with respect to a utility function.

In this work, we considered the CDNs because they are inherently working at the application layer, and thus are directly effected with the user decisions. Meanwhile, the CDNs are basically network caches and thus they reduce the network congestions and user delays. In this sense, we can measure the effects of high-level user decisions with low level network metrics such as user latency.

In the development of our results we divided the content delivery problem into distribution and routing sub-problems and investigated them separately but jointly. We showed that each sub-problem has an equilibrium given the solution of the other, and this solution is optimal if the equilibrium is unique. The distribution and routing problems appear to be similar expect that caching space is allocated in the distribution and bandwidth is allocated in the routing sub-problem. We determined the optimal strategies of the publishers and the surrogates, where each agent non-cooperatively and selfishly tries to maximize its benefit. By relying on these strategies we showed that the resulting game has an equilibrium. We further determined the uniqueness condition for the equilibrium, which appeared to be dependent on the total amount of publisher investments and the size of the resources. We noticed that if the available resource size is not large and if the publisher investments are not high, we achieve a unique equilibrium.

We also determined the system optimum solution, where the objective is to maximize the total system utility (or minimize total average user delay). We observed that the non-cooperative game representation is applicable for the distributed solution of the system optimization problem as long as the publishers are willing to pay the possibly very high investment amounts associated with this solution. Upon observing the very high prices associated with the system optimum solution, which may diminish any benefit received by using the caching architecture, we determined a sub-optimal solution where publishers select their investments to maximize their net *profit*. The publisher profit is given as the total utility received (which may be interpreted as the weighted sum of the average user delays) reduced by the total investment amount. We noticed that our sub-optimal investment strategy can increase the net benefit significantly without reducing the total system utility more than 5%.

We later discussed the joint distribution and routing game and showed that this game has an equilibrium as well. We studied the performance of the combined game as compared to current transparent proxy caching implementations numerically and observed that the improvements are more 50%.

The framework for distribution and routing is similar to an auction and allow the publishers request any range of quality of service. Meanwhile, such a quality of service may not be needed for most and only several classes of service is sufficient in many cases. We explore this possibility in the last chapter, and showed the optimal pricing schedule for this case. We also determined the condition under which the service provider chooses to offer different quality of service classes.

There remains much work to be done in this infant subject. The questions remain to be answered range fromthe design of different pricing schemes to solving implementation issues. An interesting question that can be answered in this framework is determining the optimal locations for the surrogates. Although in real networks the choices for placing a surrogate are quite limited, it is clear that a surrogate placed at a wrong location will not improve the system-wide performance any good. Our model can "move" the surrogates to the optimal locations, where their revenue is maximized.

Chapter 7

Appendices

Appendix A

Proofs of Lemmas in Chapter 3

Proof of Lemma 1

Consider the Lagrangian function $L(\mathbf{x})$,

$$L(\mathbf{x}) = \sum_{j=1}^{J} \beta_i^j \left(x_i^j\right)^{1-\alpha_i} - \gamma \left(\sum_{j=1}^{J} x_i^j p_j\right),$$

where γ is the Lagrangian constant. From Karush-Kuhn-Tucker Theorem we know that the optimal solution is given by $\partial L(\mathbf{x})/\partial x_i^j = 0$ for $\gamma \geq 0$.

$$\partial L(\mathbf{x})/\partial x_i^j = \beta_i^j (1-\alpha_i) \left(x_i^j\right)^{-\alpha_i} - \gamma p_j = 0,$$

$$x_i^j = \left(\frac{\beta_i^j (1-\alpha_i)}{\gamma p_j}\right)^{1/\alpha_i}.$$

Using this result in the constraint equation, we can determine γ as

$$\gamma^{-1/\alpha_i} = \frac{\mathcal{B}_i}{\sum_{k=1}^{J} p_k \left(\frac{\beta_i^k (1-\alpha_i)}{p_k}\right)^{1/\alpha_i}}.$$

Now, optimal x_i^j can easily be determined. ■

Proof of Lemma 2

As illustrated in Figure 3.4, the surrogate's revenue decreases when the price is either increased or decreased beyond a certain price. Let $r_j(p_j)$ denote the revenue of surrogate j.

$$r_j(p_j) = \begin{cases} \sum_i x_i^j(p_j)p_j & \text{for } \sum_i x_i^j(p_j) \leq C_j \\ C_j p_j & \text{for } \sum_i x_i^j(p_j) > C_j \end{cases}.$$

The optimal point lies either at the boundaries or at the irregularity. If $\sum_i x_i^j(p_j) \leq C_j$,

$$\begin{aligned} \partial r_j/\partial p_j &= \mathcal{B}_i \frac{(\beta_i^j)^{1/\alpha_i}(-1/\alpha_i)p_j^{-1-1/\alpha_i}\sum_k p_k^{1-1/\alpha_i}(\beta_i^k)^{1/\alpha_i}}{\left(\sum_k p_k^{1-1/\alpha_i}(\beta_i^k)^{1/\alpha_i}\right)^2} \\ &\quad -\mathcal{B}_i \frac{(\beta_i^j)^{1/\alpha_i}p_j^{-1/\alpha_i}(1-1/\alpha_i)(\beta_i^j)^{1/\alpha_i}p_j^{-1/\alpha_i}}{\left(\sum_k p_k^{1-1/\alpha_i}(\beta_i^k)^{1/\alpha_i}\right)^2} \\ &= \mathcal{B}_i \frac{-(\beta_i^j)^{2/\alpha_i}p_j^{-2/\alpha_i} - 1/\alpha_i(\beta_i^j)^{1/\alpha_i}p_j^{-1-1/\alpha_i}\sum_{k\neq j} p_k^{1-1/\alpha_i}(\beta_i^k)^{1/\alpha_i}}{\left(\sum_k p_k^{1-1/\alpha_i}(\beta_i^k)^{1/\alpha_i}\right)^2} \\ &< 0. \end{aligned}$$

Thus, $r_j(p_j)$ is monotonically decreasing for p_j such that $\sum_i x_i^j(p_j) \leq C_j$.

If $\sum_i x_i^j(p_j) > C_j$, then it is clear that $r_j(p_j)$ is maximized at the boundary when $\sum_i x_i^j(p_j) = C_j$. This concludes the proof. ■

Proof of Lemma 3

$$\begin{aligned} \sum_i \frac{{p_j}^{-1/\alpha}}{\sum_k p_k^{1-1/\alpha}}\mathcal{B}_i &= C_j \\ {p_j}^{-1/\alpha}\left(1 - \frac{C_j}{\sum_i \mathcal{B}_i}p_j\right) &= \frac{C_j}{\sum_i \mathcal{B}_i}\sum_{k\neq j} p_k^{1-1/\alpha} \qquad \text{(A.1)} \end{aligned}$$

Let $\gamma_j = \sum_i \mathcal{B}_i / C_j$. Taking the derivative of eq. (A.1) with respect to p_l, we determine $\partial p_j / \partial p_l$.

$$
\begin{aligned}
-\frac{1}{\alpha p_j} p_j^{-1/\alpha} \frac{\partial p_j}{\partial p_l}(1 - p_j/\gamma_j) - \frac{1}{\gamma_j} p_j^{-1/\alpha} \frac{\partial p_j}{\partial p_l} &= \frac{1}{\gamma_j}(1 - 1/\alpha) p_l^{-1/\alpha} \\
\frac{\partial p_j}{\partial p_l} = \frac{p_l^{-1/\alpha}}{p_j^{-1/\alpha}\left(\frac{\gamma_j}{(1-\alpha)p_j} - 1\right)} &= \frac{(1-\alpha)\frac{p_j}{\gamma_j} p_l^{-1/\alpha}}{p_j^{-1/\alpha}\left(1 - (1-\alpha)\frac{p_j}{\gamma_j}\right)} \qquad \text{(A.2)}
\end{aligned}
$$

Notice that the denominator in eq. (A.2) is similar to the left hand side of eq. (A.1). Remember that $0 < \alpha < 1$. Hence, $p_j^{-1/\alpha}\left(1 - (1-\alpha)\frac{p_j}{\gamma_j}\right) > p_j^{-1/\alpha}\left(1 - \frac{p_j}{\gamma_j}\right)$. Then,

$$
\begin{aligned}
\frac{\partial p_j}{\partial p_l} &< \frac{(1-\alpha)\frac{p_j}{\gamma_j} p_l^{-1/\alpha}}{p_j^{-1/\alpha}\left(1 - (1-\alpha)\frac{p_j}{\gamma_j}\right)} \\
&= \frac{(1-\alpha) p_j p_l^{-1/\alpha}}{\sum_{k \neq j} p_k^{1-1/\alpha}}
\end{aligned}
$$

Also notice that $p_j < \gamma_j$. Then $0 < 1 - p_j/\gamma_j < 1$. From eq. (A.1)

$$
\begin{aligned}
p_j^{1/\alpha} &= \frac{1 - p_j/\gamma_j}{\frac{1}{\gamma_j}\sum_{k \neq j} p_k^{1-1/\alpha}} \\
&< \frac{1}{\frac{1}{\gamma_j}\sum_{k \neq j} p_k^{1-1/\alpha}} \\
p_j &< \frac{\gamma_j^\alpha}{\left(\sum_{k \neq j} p_k^{1-1/\alpha}\right)^\alpha}
\end{aligned}
$$

Thus, the best response function $R_j(\mathbf{p})$ is a contraction mapping if

$$
\frac{\partial p_j}{\partial p_l} < \frac{(1-\alpha)\gamma_j^\alpha p_l^{-1/\alpha}}{\left(\sum_{k \neq j} p_k^{1-1/\alpha}\right)^{1+\alpha}} < 1
$$

■

Appendix B

Near-optimal publisher investment strategy

In this appendix, we show the derivation of the near-optimal investment strategy. For the sake of simplicity we determine our results for $\alpha_i = 0.5, \forall i$.

For a given investment B_i, the optimal cache allocation is given by

$$x_{ij}(B_i) = \frac{\beta_{ij}^2/p_j^2}{\sum_k \beta_{ik}^2/p_k} B_i.$$

Assume that p_j are the set of prices at the equilibrium. As for the key assumption of our derivation, we assume that for a small change in investment amounts, the change in the equilibrium prices is small as well. Specifically, in the vicinity of the set of investment amounts B_i, $\sum_k \beta_{ik}^2/p_k$ can be considered as *constant*. This assumption is verified through numerical studies.

Let $\sum_k \beta_{ik}^2/p_k = h_i$. Thus, at the equilibrium the surrogate prices are,

$$p_j = \sqrt{1/C_j \sum_i \beta_{ij}^2/h_i B_i},$$

and the cache allocation is

$$\frac{\beta_{ij}^2 B_i}{h_i p_j^2}.$$

We are interested in the optimal investments maximizing the net profit, i.e. $\sum_j x_{ij}^{0.5}\beta_{ij} - B_i/T$. Then,

$$\sum_j 1/2 x_{ij}^{-1/2}(B_i)\frac{\partial x_{ij}}{\partial B_i}\beta_{ij} - 1/T = 0,$$

where

$$\frac{\partial x_{ij}}{\partial B_i} = \frac{\beta_{ij}^2}{h_i}\left(\frac{1}{p_j^2} - \frac{2B_i}{p_j^3}\frac{\partial p_j}{\partial B_i}\right),$$

and

$$\begin{aligned}\frac{\partial p_j}{\partial B_i} &= 1/2\left(1/C_j\sum_i \beta_{ij}^2/h_i B_i\right)^{-1/2} 1/C_j\beta_{ij}^2/h_i \\ &= 1/2\frac{\frac{\beta_{ij}^2}{\sqrt{C_j\sum_k \beta_{ik}^2/p_k}}}{\sqrt{\sum_l \frac{\beta_{lj}^2 B_l}{\sum_k \beta_{lk}^2/p_k}}}\end{aligned}$$

Then, the optimal investment can be calculated by the simultaneous solution of the following set of nonlinear equations.

$$\sum_j 1/2\beta_{ij}\sqrt{B_i}\left(\frac{\beta_{ij}^2/p_j^2}{\sum_k \beta_{ik}^2/p_k}\right)^{3/2}\left[1 - \frac{B_i\beta_{ij}^2/p_j^2}{C_j\sum_k \beta^2{}_{ik}/p_k}\right] = 1/T,$$

and

$$\sum_i \frac{\beta_{ij}^2/p_j^2}{\sum_k \beta^2{}_{ik}/p_k}B_i = C_j, \forall j.$$

Let $\gamma_{ij} = \frac{\beta_{ij}^2/p_j^2}{\sum_k \beta^2{}_{ik}/p_k}$, and $U_i = \sqrt{B_i}$, then the optimal solution is the solution to the following much simpler nonlinear equation:

$$1/T - 1/2U_i\sum_j \beta_{ij}\gamma_{ij}^{3/2} + 1/2U_i^3\sum_j \frac{\beta_{ij}\gamma_{ij}}{C_j} = 0, \tag{B.1}$$

and $\sum_i \gamma_{ij} B_i = C_j, \forall j$.

A simple near-optimal investment strategy determines the investment amount B_i from Eq.(B.1) for the equilibrium prices.

Bibliography

[1] J. Almeida, M. Dabu, A. Manikutty and P. Cao, " Providing Differentiated Levels of Service in Web Content Hosting," *Proceedings of ACM SIGMETRICS Workshop on Internet Server Performance*, 1998.

[2] L. Amini, S. Thomas and O. Spatscheck, "Distribution Peering Requirements for Content Distribution Internetworking," (work in progress) draft-amini-cdi-distribution-reqs-00.txt, 02/2001.

[3] L. Anania and R. J. Solomon, "Flat:The Minimalist B-ISDN Rate", pp.91-118 in *Internet Economics*, MIT Press, 1997.

[4] H. Bahn, S. H. Noh, S. L. Min, K. Koh, "Using Full Reference History for Efficient Document Replacement in Web Caches," *Proceedings of USENIX Symp. on Internet Tech. and Systems*, pp 187-196, 1999.

[5] A. Banchs, O. Leon and S. Sallent, "The Olympic Service Model: Issues and Architecture," Workshop on Quality of Service of the Future Internet, 2001.

[6] A. Barbir, B. Cain, M. Green, M. Hofmann, R. Nair, B. Potter and O. Spatscheck, "Known CDN Request-Routing Mechanisms," (work in progress) draft-cain-cdnp-known-request-routing-02.txt, 06/2001.

[7] C. Berge, *Topological Spaces*, Macmillan, New York, 1963.

[8] D. Bertsekas, *Nonlinear Programming*, 2nd ed., Athena Scientific, 1999.

[9] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice-Hall, 1989.

[10] A. Bestavros, "Demand-based document dissemination to reduce traffic and balance load in distributed information systems", *Proceedings of SPDP'95*

[11] A. Biliris, C. Cranor, F. Douglas, M. Rabinovich, S. Sibal, O. Spatscheck and W. Sturm, "CDN Brokering," Proceedings of WCW'01, June 2001.

[12] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "RFC 2475: An Architecture for Differentiated Services," *IETF Networking Group Request for Comments*, http://www.ietf.org/rfc/rfc2475.txt.

[13] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications", *Infocom'99*.

[14] B. Cain, O. Spatscheck, M. May and A. Barbir, "Request-Rounting Requirements for Content Internetworking," (work in progress) draft-cain-request-routing-req-01.txt, 01/2001.

[15] P. Cao and S. Irani, "Cost-aware www Proxy caching Algorithms," *USENIX Symposium on Internet Tech. and Systems*, pp. 193-206, 1997.

[16] P. Chander and L. Leruth, "The Optimal Product Mix for a Monopolist in the Presence of Congestion Effects," *International Journal of Industrial Organization*, v. 7, p.p. 437-449, North-Holland, 1989.

[17] S. Chen and K. Nahrstedt, "An overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," *IEEE Network*, pp. 64-79, Nov./Dec. 1998.

[18] J. Chuang and M. Sirbu, "stor-serv:Adding Quality of Service to Network Storage," *Workshop on Internet Service Quality Economics*, 1999.

[19] J. Chuang, "Resource Allocation for stor-serv: Network Storage Services with QoS Guarantees," *NetStore'99*, Seattle WA, Oct 1999.

[20] M. Cieslak, D. Forster, G. Tiwana, and R. Wilson, "Web Cache Coordination Protocol v2.0," *Internet Draft*, expires Jan 2001.

[21] R. Cocchi, D. Estrin, S. Shenker, and L. Zhang, "Pricing in Computer Networks: Motivation, Formulation and Example," *IEEE/ACM Transactions on Networking*, 1:614-627, 1993.

[22] I. Cooper, J. Dilley, "Known HTTP Proxy/Caching Problems," Internet Draft, expires Jan 11, 2001.

[23] C. Courcoubetis, G. D. Stamoulis, C. Manolakis and F. P. Kelly, "An Intelligent Agent for Optimizing QoS-for-Money in Priced ABR Connections," to appear in *Telecommunications Systems, Special Issue on Network Economics* available at http://www.ics.forth.gr/proj/race/publications/abr_agent.html

[24] P. B. Danzig, R. S. Hall, and M. F. Schwartz, "A Case for caching file objects inside internetworks." *Proceedings of SIGCOMM'93*, pp 239-248, 1993.

[25] M. Day, B. Cain, G. Tomlinson and P. Rzewski, "A Model for content Internetworking," (work in progress) draft-day-cdnp-model-05.txt, 03/2001.

[26] L. Fan, P. Cao, J. Almeida, A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *Technical Report 1361*, Department of Computer Science, University of Wisconsin-Madison, Feb 1998.

[27] P. Ferguson and G. Huston, *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, Wiley, 1998.

[28] C. Fricker and M. R. Jai, "Monotonicity and stability of periodic polling models," RR-1690, INRIA, citeseer.nj.nec.com/fricker94monotonicity.html

[29] E. W. Fulp, M. Ott, D. Reininger and D. S. Reeves, "Paying fofr QoS: An Optimal Distributed Algorithm for Pricing Network resources," *Proceedings of the IEEE 6th IwQoS*, pp. 75-84, 1998.

[30] J. J. Gabszewicz, A. Shaked, J. Sutton and F. Thisse, "Segmenting the Market: The Monopolist's Optimal Product Mix," *Journal of Economic Theory*, vol. 39, no. 2, pp. 273-289, 1986.

[31] P. Gauthier, J. Cohen, M. Dunsmuir, C. Perkins, "The Web Proxy Auto-Discovery Protocol", Internet Draft, Oct 1998

[32] R. J. Gibbens and F. P. Kelly, "Resource Pricing and the Evolution of Congestion Control," *Automatica*, v.35, 1999.

[33] R. Gibbens, R. Mason and R. Steinberg, "Internet Service Classes under Competition," *IEEE Journal on Selected Areas in Communications*, v.18 no.12, p.p. 2490-2498, Dec 2000.

[34] M. Green, B. Cain, G. Tomlinson, S. Thomas and P. Rzewski, "Content Internetworking Architectural Overview," (work in progress) draft-green-cdnp-gen-arch-03.txt, 03/2001.

[35] A. Gupta, D. O. Stahl and A. B. Whinston, "Managing the Internet as an Economic System," *Technical Report*, Univ. of Texas, 1994.

[36] A. Gupta, D. O. Stahl, and A. B. Whinston, "The Economics of Network Management," *Communications of the ACM*, v. 42, no.9, Sept. 1999.

[37] J. Gwertzman and M. Seltzer, "World Wide Web Cache Consistency," *Proceedings of 1996 USENIX Conference*, San Diego, CA, Jan 1996.

[38] D. Hazlett, "Congestion Pricing," *preprint*. available at ftp://marcus.whitman.edu/pub/Hazlett/congestion.ps

[39] J. Heinan, F. Baker, W. Weiss and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.

[40] L. Hurwicz, "The Design of Mechanisms for Resource Allocation," *American Economic Review*, 63(2):1-30, 1973.

[41] G. Alexander Jehle, *Advanced Microeconomic Theory*, Prentice-Hall, 1991.

[42] J. Kangasharju and K.W. Ross, "Replicating the DNS using Satellite Broadcast," *Infocom 2000*.

[43] J. Kangasharju, K. W. Ross, and J. W. Roberts, "Performance Evaluation of Redirection Schemes in ontent Distribution Networks," *5th Web Caching Workshop*, Lisbon, Portugal, May 22 - 24, 2000.

[44] M. Karaul, Y. A. Korilis, and A. Orda, "A Market-Based Architecture for Management of Geographically Dispersed, Replicated Web Servers," *Proceedings of the First International Conference on Information and Computation Economies (ICE)*, Charleston, SC, October 1998, pp. 158 - 165.

[45] R. Karedla, J.S. Love, B. G. Wherry, "Caching Strategies to Improve Disk System Performance," *IEEE Computer*, pp. 38-46, v.27, n.3, 1994

[46] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin and R. Panigrahy, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," Proceedings of STOC 97, El Paso, TX, 1997.

[47] Frank Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, v.8 pp 33-37, 1997.

[48] F. Kelly, A. Maulloo and D. Tan, "Rate Control in Communications Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society* , vol. 49, pp. 237-252, 1998.

[49] T. Kelly, Y. Chan, S. Jamin and J. MacKie-Mason, "Biased Replacement Policies for Web Caches: Differential QoS and Aggregate User Value," *Proceedings of 4th International Web Caching Workshop*, 1999.

[50] T. P. Kelly, S. Jarmin and J. K. MacKie-Mason, "Variable QoS from Shared Web Caches: User-Centered Design and Value Sensitive Replacement," *MIT workshop on Internet Service Quality Economics*, Dec'1999.

[51] T. P. Kelly, S. Jarmin and J. K. MacKie-Mason, "Variable QoS from Shared Web Caches: User-Centered Design and Value Sensitive Replacement," *MIT workshop on Internet Service Quality Economics*, Dec'1999.

[52] T. Kelly, and D. Reeves, "Optimal Web Cache Sizing: Scalable Methods for Exact Solutions," *Fifth International Web Caching and Content Delivery Workshop*, Lisbon, Portugal, May 2000.

[53] Y. A. Korilis and A. A. Lazar, "On the Existence of Equilibria in Noncooperative Optimal Flow Control," *Journal of the ACM*, vol. 42, no. 3, pp. 584-613, May 1995.

[54] J. F. Kurose, K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, Addison-Wesley, 2001.

[55] J. F. Kurose and R. Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems," *IEEE Trans. on Computers*, 38(5):705-717, 1989.

[56] B. Li, X. Deng, M. Golin, and K. Sohraby, "On the optimal placement of web proxies in the internet", *INFOCOM'99*, pp 1282-1290

[57] T. Liao. "WebCanal: Multicast Based Push," W3C Push Workshop, Boston, 1997

[58] C. Liu, and P. Cao, "Maintaining Strong Cache Consistency in the Worl-Wide Web," *Proceedings of ICDSC'97*, pp. 12-21, May 1997.

[59] B. Liver and G. Dermler, "The E-Business of Content Delivery," *International Workshop on Internet Service Quality Economics*, MIT, Cambridge, Massachusetts, U.S.A., December 2-3, 1999.

[60] P. Lorenzetti, L. Rizzo, and L. Vicisano, "Replacement Policies for a Proxy Cache," *Tecnical Report UCL-CS Research Note RN/98/13*, 1998.

[61] N. Van Long and A. Soubeyran, "Existence and uniqueness of Cournot Equilibrium: A contraction mapping approach," *Economics Letters*, vol 67 (2000), pp 345-348, Elsevier Publishing.

[62] S. H. Low and D. E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol 7 no 6, Dec. 1999.

[63] David G. Luenberger, *Microeconomic Theory*,McGrawHill, 1994.

[64] J. K. MacKie-Mason and H. Varian, "Pricing the Internet," in *Public Access to the Internet*, eds. B. Kahin and J. Keller, Prentice-Hall, 1995.

[65] J. K. MacKie-Mason and H. Varian, "Pricing Congestible Network Resources," *IEEE Journal of Selected Areas in Communications*, vol.13, no. 7 (Sept. 1995): 1141-49.

[66] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd and V. Jacobson, "Adaptive Web Caching: Towards a New Global Caching Architecture," *3rd International WWW Caching Workshop*, June 1998.

[67] B. M. Mitchell and I. Vogelsang, "Telecommunications Pricing: Theory and Practice," p.p. 73-74, Cambridge University Press, 1991.

[68] M. Nottingham, "Requirements for Demand Driven Surrogate Origin Servers," (work in progress) draft-nottingham-surrogates-00.txt, 01/2001.

[69] A. M. Odlyzko, "Paris Metro Pricing: The minimalist differentiated services solution", Proc. IWQoS'99, pp. 159-161.

[70] P. Rodriguez, E.W. Biersack and K.W. Ross, "Automated Delivery of Web Documents Through a Caching Infrastructure," preprint.

[71] P. Rodriguez, A. Kirpal, and E. W. Biersack, "Parallel-Access for Mirror Sites in the Internet," *Proc. of Infocom 2000*, Tel-Aviv, Israel, March 2000.

[72] P. Rodriguez, E.W. Biersack and K.W. Ross, "Automated Delivery of Web Documents Through a Caching Infrastructure," *preprint*.

[73] A. Rousskov, D. Wessels, "Cache Digests", NLANR, April 1998

[74] H. Salama, D. S. Reeves and Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing," Proceedings of Infocom 1997.

[75] P. Scheuermann, J. Shim, and R. Vingralek, "A Case for delay-conscious caching of web documents," *Proceedings of the 6th International WWW Conference*, 1997.

[76] S. Shenker, D. Clark, D. Estrin and S. Herzog, "Pricing in Computer Networks: Reshaping the Research Agenda," *Journal of Telecommunications Policy*, v.20, no.3, pp. 183-201, 1996.

[77] P. Thomas, D. Teneketzis, and J. K. MacKie-Mason, "A Market-Based Approach to Optimal Resource Allocation in Integrated-Services Connection-Oriented Networks," *preprint*.

[78] J. Touch, "The LSAM Proxy Cache: A Multicast Distributed Virtual Cache," *3rd International WWW Caching Workshop*, Manchester, UK, June 15-17, 1998.

[79] J. N. Tsitsiklis and I. C. Paschalidis, "Congestion-Dependent Pricing of Network Services," *IEEE/ACM Transactions on Networking* v.8 no.2, pp. 171-184, 2000.

[80] H. Varian and J. K. MacKie-Mason, "Genralized Vickrey Auctions," *Technical Report, Dept. of Economics*, U. Mich, 1994.

[81] William Vickrey, "Counterspeculation, auctions and competitive sealed tenders," *Journal of Finance*, 16:8-37, 1961.

[82] Santiago J. Villasis, "An Optimal Pricing Mechanism for Internet's End-Users," *MS Thesis*, Economics Dep., Univ. of Idaho, 1996. available at http://www.uidaho.edu/ econ/svthes.pdf

[83] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E.Fox, "Removal policies in network caches for www documents," *Proceedings of ACM SIGCOMM'96*, pp 293-305, 1996.

[84] R. Wooster and M. Abrams, "Proxy caching that estimates page load delays," *Proceedings of 6th International WWW Conference*, 1997.

[85] H. Yaiche, R. Mazumdar and C. Rosenberg, "A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667-678, 2000.

[86] http://www.index.berkeley.edu

[87] H. Yen and F. Y. Lin, "Near-optimal Delay Constrained Routing in Virtual Circuit Networks," Proceedings of Infocom 2001.

[88] G. K. Zipf *Human Behavior and the Principle of Least Effort*, Addison-Wesley Press, Cambridge MA, 1949.

[89] Zona Research, "Economic Impact of Unacceptable Web Site Download Speeds"

[90] http://ircache.nlanr.net

[91] http://www.microsoft.com/Proxy/Guide/CarpWP.asp

[92] http://www.alteonwebsystems.com/products/white_papers/DNS/index.shtml

[93] http://www.akamai.com

[94] http://www.mirror-image.com

[95] http://www.digitalisland.com