1-1-2005

Traffic engineering friendly request routing in content delivery network

Arun Nanthakumaran
Ryerson University

Follow this and additional works at: <http://digitalcommons.ryerson.ca/dissertations>

Part of the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Nanthakumaran, Arun, "Traffic engineering friendly request routing in content delivery network" (2005). *Theses and dissertations*. Paper 364.

This Thesis is brought to you for free and open access by Digital Commons @ Ryerson. It has been accepted for inclusion in Theses and dissertations by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact bcameron@ryerson.ca.

TRAFFIC ENGINEERING FRIENDLY REQUEST ROUTING IN CONTENT DELIVERY NETWORK

By

Arun Nanthakumaran

**B. Sc. Eng in Electronics & Telecommunications
University of Moratuwa, Sri Lanka, April 1999**

**A thesis presented to Ryerson University in partial fulfillment of
the requirements for the degree of**

Master of Applied Science in Electrical and Computer Engineering

Toronto, Ontario, Canada, 2005

©Arun Nanthakumaran 2005

UMI Number: EC53746

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform EC53746
Copyright 2009 by ProQuest LLC
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Author's Declaration Page

I hereby declare that I am the sole author of this thesis or dissertation. I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

Signature: _____

September 2005.

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature: _____

September 2005

ABSTRACT

**Arun Nanthakumaran, Traffic Engineering Friendly Request Routing in CDN,
Electrical and Computer Engineering, Ryerson University, 2005.**

Content delivery over Internet is optimized for low user perceived latency by designing specialized Content Distribution Network (CDN). Content servers are connected through a content switch that distributes client requests among them to achieve load balancing across the servers. The content switch is located one-hop away from the servers. Server load balancing is one factor in achieving low user perceived latency and improving operational efficiency. Traffic engineering is another factor that is generally not integrated with server load balancing for CDN optimization. In this thesis we propose a request routing algorithm for a CDN that was designed to integrate server selection and traffic engineering functions in the request routing system. The CDN employs MPLS in the network for traffic engineering. The proposed algorithm optimizes content delivery for user perceived latency by achieving server load balancing and network traffic load management among alternative paths. It also improves operational efficiency of the CDN by eliminating bottleneck paths and increasing utilization of underutilized servers and paths.

Key words:

Content Delivery Networks, Traffic Engineering, Server load balancing, Request routing.

A C K N O W L E D G E M E N T S

I would like to thank my supervisor Dr. Muhammad Jaseemuddin for his support, guidance, and knowledge and giving me the opportunity to be a member of the MOWIN Laboratory. I want to acknowledge all the members of this lab for their help and for being both great friends and colleagues.

My special thanks to Dr. S. Krishnan, Dr. Isaac Woungang and Dr. Bobby Ma for their valuable comments and for being on the defense committee

I would also like to thank my parents, my wife for their great support and endless love during my study period. Without you all it would be impossible to achieve this milestone.

TABLE OF CONTENTS

Author's Declaration Page	ii
Abstract	iii
Acknowledgements	iv
List of Tables	vii
List of Figures	viii
List of Appendices	ix
1. Introduction	01
2. Background & Related Work	
2.0 Introduction	04
2.1 Server Selection Methods	06
2.2 IP Anycasting	08
2.3 MPLS Traffic Engineering	11
2.4 Bandwidth Guaranteed Algorithms	14
2.5 Request Routing Algorithms	16
2.6 Chapter Summary	20
3. Network Design Architecture	
3.0 Introduction	21
3.1 The CDN Architecture for TFRR	22
3.2 Design Issues	24
3.3 Network Design Solution	26
3.4 Chapter Summary	29
4. TE Friendly Request Routing Algorithm	
4.0 Introduction	30
4.1 Problem Definition	31

4.2	Server Load Metrics	33
4.3	Path Load Metrics	35
4.4	Selection Process	37
4.5	The Algorithm	40
4.6	Chapter Summary	44
5.	Simulation & Results	
5.0	Introduction	45
5.1	Workload Characterization	45
5.2	Network Simulators	47
5.3	Simulation using NS-2	47
5.4	Simulation using OPNET	50
5.5	Chapter Summary	69
6.	Conclusion	70
Appendix 1		72
Appendix 2		73
Appendix 3		76
Appendix 4		83
References		88

LIST OF TABLES

Table #	Description	Page #
5.1	Workload parameters	46
5.2	Traffic load matrix	55
5.3	Demand weight table	56
5.4	Critical weight table	57
5.5	Average weight table	58
5.6	Path weight table	58
5.7	Path selection for [TFRR+RR] model	59
5.8	Expected load of client network	60
5.9	Expected load on servers	61
5.10	Average weight table – [TFRR+RI] model	61
5.11	Path weight table – [TFRR+RI] model	62
5.12	Server selection and path selection	63

LIST OF FIGURES

Figure #	Description	Page #
2.1	MPLS Network Architecture	11
2.2	Architecture of a TE enable LER node	14
3.1	The CDN architecture employing TFRR to balance server load and network path load	22
3.2	Design of CDN employing TFRR	26
4.1	Network Diagram	31
5.1	Simulation Topology	48
5.2	Response time of web requests for (a) proportional load distribution and (b) round robin load distribution	49
5.3	Load distribution under Round robin scheme and Propotional bandwidth scheme.	50
5.4	The profile configuration	52
5.5	Simulation Network Topology – Conventional CDN	52
5.6	Client subnets	53
5.7	Client Network	53
5.8	Simulation Network Topology – Proposed CDN	55
5.9	Network diagram	56
5.10	Link Load Statistics on ingress links in bps	64
5.11	Link Load Statistics on egress links in bps	64
5.12	Link Load Statistics on intermediate links in bps	64
5.13	Server Load Statistics bytes/sec	65
5.14	Simulation model	67
5.15	Link load statistics on ingress links in bps	68
5.16	Queuing delay of ingress links in seconds	68

LIST OF APPENDICES

Appendix #	Description	Page #
1	Traffic Load per Client (Bps)	72
2	Path Selection for Proposed CDN [TFRR + RR] Model	73
3	Server Selection and Path Selection for Proposed CDN [TFRR + RI] model	76
4	Probability Distributions	83

1

Introduction

Content delivery over Internet is growing and becoming more demanding on the network resources. Search engines, multi media streaming, business applications and application services access content through web interface. The growing number of content requests cannot only be served with a single web server. Geographically distributed multiple web servers are required to respond the content requests quickly and with desired Quality of Service (QoS). Hence, content routing has recently gained significant attention. Load balancing among the similar content servers become a critical issue, while maintaining the network path load distribution.

Domain Name System (DNS) selects an IP address of a server among a set of servers hosting the same content in a round-robin fashion to achieve server load balancing. Although it is by far the most widely used mechanism, it is ineffective as it never takes into account QoS parameters such as delay, bandwidth. Content switches that perform Layer 4 to layer 7 classifications and forwarding are deployed close to a server farm for load balancing among the servers. They can achieve server load balancing for only among local servers. Global load balancing over identical servers geographically distributed may not be achieved since each context switch relies on DNS based round-robin selection among the distributed servers. IP Anycasting can be used to make effective server load balancing over geographically distributed servers. IP Anycasting has recently become an attractive communication paradigm in which servers share a single IP anycast address and the request is delivered to only one of the anycast domain servers that is closest to the client.

Balancing server load is not enough to provide optimum network performance. The content should be routed through a network path to effectively distribute the network path load among all the available paths in the network. To achieve network load distribution,

the network should be enabled with Traffic Engineering (TE) capabilities. Traditional IP backbones have limitations in applying TE principles. We leverage the growing migration of networks to Multi Protocol Label Switching (MPLS) to provide flexible routing and the ability to perform network traffic engineering.

The objective of this thesis is to propose a request routing algorithm that performs: (1) server selection to select the least loaded closest server, and (2) to select the path inside the network for the content delivery that distributes the network load. The algorithm is to manage both server load distribution and network path load distribution to satisfy the client with the desired quality of service.

The thesis is organized as follows. Chapter 2 discusses the background and related work. It describes the conventional methods used in providing load balancing among multiple web servers that are already in use with their pros and cons. The IP Anycasting technique is identified as a potential technique and described in detail. Various issues related to IP Anycasting method are also explored with the remedial suggestions. While IP Anycasting is used to achieve server load balancing, MPLS traffic engineering will be applied to achieve network path load distribution. The MPLS facilitates TE by using the Constraint Based Routing (CBR) rather than traditional routing methods. The CBR mechanism and how the MPLS architecture supports it are analyzed in the subsection of MPLS Traffic Engineering. In addition, most commonly used bandwidth guaranteed TE algorithms are described, ranging from simple Widest Shortest Path (WSP) scheme to complex Minimum Interference Routing Algorithm (MIRA). Some noted request routing algorithms that achieve either server load balancing or network load distribution or both are also explained.

In Chapter 3, a Content Delivery Network (CDN) architecture that supports the achievement of server load balancing and network path load distribution simultaneously is proposed. The backbone architecture is based on MPLS network to enable fast switching of packets through the label switched paths and facilitate Traffic Engineering. Clients send request packets using anycast address as the destination server address. All

packets from the clients are sent with anycast address in the proposed. This chapter explains the architectural design and issues arise and how those issues are resolved.

Chapter 4 proposes a TE friendly Request Routing (TFRR) algorithm to achieve server load balancing and network load distribution. The algorithm considers residual server load index, average path weight based on interference, criticality and utilization when making a selection of a path and a content server. In Chapter 5, workload characterization is briefly explained and how a workload model can be applied in a network simulation. The initial simulation done using NS-2 software to assess the feasibility of our approach is discussed. A detailed simulation model is designed and explained using OPNET modeler software. The TFRR algorithm is tested using OPNET simulation tool. Network and Server load performances are compared between conventional CDN models, proposed TFRR with conventional round-robin server selection, and the proposed TFRR with its own server selection process. Finally, Chapter 6 discusses the conclusion and future work.

2

Background & Related Work

2.0 INTRODUCTION

Content servers in a Content Delivery Network (CDN) provide content to wide variety of users, and should have the ability to handle more number of client requests concurrently. However, the server should not be overloaded by allowing more number of connections than it can handle without degrading the performance. With a growing number of web content access over the Internet, it is not feasible to keep the web contents in only one server. A single server cannot handle all the requests simultaneously and it is limited by its capacity. Further, the client response time cannot be kept low in providing the service. A feasible solution to these problems is to replicate content over multiple servers located in a wide geographical area, so that more requests can be handled and the response time can be lowered by selecting the server that is closest to the client location.

Client access time can be improved by controlling the server response time. Load balancing across multiple servers is an effective way to decrease the server response time. There are various methods employed to access web resources and keeping the server load to a minimum. When there are multiple servers with similar content are setup, a proper server selection algorithm should be applied to achieve load balancing among the content servers. However, achieving server load balancing is a challenging issue.

One of the earliest ideas was caching the web server content in the client web browser or in the proxy servers that are located close to the clients. These approaches are still popular for sites with more images and search engines. If previously requested data was requested by another user, the content would be delivered locally rather than contacting the server which is in the remote site. Internet Service Providers (ISPs) and Content

Delivery Network (CDN) operators usually keep cache servers to serve web requests locally instead of utilizing international backbone bandwidth that is more expensive. This method is not appropriate for more dynamic content of data and some sites or types of data that cannot be cached. However user behavior is normally to get 'fresh' content rather than cached content. Thus, ISPs design their proxies as transparent proxies that are transparent to the users.

In another method, the web contents are kept in mirrored sites by storing the similar content in 'mirrored' web servers distributed over wide geographical locations. Each server is configured with different IP addresses. The client has the responsibility to choose the closest proximity server to get a faster response. But the client has to spend sometime to choose the suitable server, and this manual selection process may take too long. The mirrored sites are updated frequently; otherwise some users will get the out-dated information. Maintaining a perfect synchronization between mirror sites may not be an easy job, particularly for time critical applications. A CDN operator mainly focuses on these types of services to generate more revenue.

In an alternative method, a cluster of web servers in a single location called server farm can be used to share the server load and increase the web server availability. The server farm can be setup with either replicated content or distributed content. This method avoids user involvement in selecting a server. However, this method requires an additional hardware called 'dispatcher' to make the decision of selecting the server. All clients request are sent to the dispatcher, which then forwards the requests to different web servers based on various load balancing algorithms placed on it. The main drawback is that the setup is subject to a single point of failure at the dispatcher. Another drawback is that the dispatcher is too expensive. Cisco's Local Director, Altheon's ACE Director are some examples of dispatchers available in the industry. These dispatchers are generally TCP layer switches where the dispatcher examines the TCP header once it receives a packet and forward the packet based on the TCP header information. There are switchers that work at application layer level where they examine the http headers to forward the packets.

When multiple servers with same type of contents are employed, the closest server should be selected to serve the client request, at the same time the server should be selected in such a way to evenly distribute the load among all available servers. When a server farm is employed, the client proximity is same for all the servers in the server farms so the dispatcher should select the proper server that is lightly loaded. There are several server selection methods available.

2.1 SERVER SELECTION METHODS

The server load distribution mainly depends upon the server selection mechanism. The selection scheme should focus on improving client performance. In addition to the main goal, the selection method should support a rich and flexible set of selection criteria.

Server selection methods are either client based or server based. In client-based approaches, the client has to determine the proper server. It could be based on the user's web browser, but that option is only for the companies that have the direct control over the browser software. An increasingly popular and simple technique for directing clients to a nearest server is to perform the server selection during name resolution using a Domain Name System (DNS) server. When several IP addresses are associated with a web site, the resolver can choose one IP address of the web site. The DNS server typically uses a round robin scheme to allocate the servers to the clients. The transparent nature of name resolution can be exploited to redirect clients to an appropriate server without any modifications to client software, server protocols or web applications. The main advantages of the DNS based method are its simplicity and that it requires no modification to the existing network. However, client side caching of server IP address can lead to improper load distribution among the content distribution servers. There are several algorithms exist to reduce TTL value of the resource records to minimize the effect of caching DNS entries. This in turn increases the name resolution latency. Another important issue is that DNS method assumes that clients and their local name servers are proximal. A. Shaikh et al [18] explored these two issues related to the DNS based server selection. DNS schemes typically disable the client side caching of name

resolution and this leads to higher latency in name resolutions. And DNS based server selection always relies on the proximity of clients with the name servers. However, this rule is grossly violated according to their study. Further, changes in DNS information propagate slowly across the Internet. Even after a detection of failure of DNS server and removing its records, the whole Internet may not become aware this effect for hours or possibly for number of days. As a result, the failed server continues to get requests and those requests will eventually fail. And the other major issue with DNS based selection is that DNS servers fail to consider network topology and current load in making decisions. Consequently, a client may be directed to a far away web server or a server attached to a congested network even though another server is available with close proximity.

Dispatchers are special switches or routers that forward the requests to different web servers based on a load balancing algorithm. They distribute TCP connections over multiple servers based on hop count or link latency between server and client. They support up to millions of connections simultaneously at high speeds. Further, they frequently check the servers load conditions so that they make some intelligent decisions regarding load balancing. They normally work at TCP layer; hence it is difficult to design algorithms with HTTP object and Quality of Service (QoS) functionalities. An application level switch is required to explore http headers to make the load balancing decisions. The main drawback of this scheme is that the dispatcher becomes a single point of failure in the network. It can be very expensive to keep an identical dispatcher as a backup in the event of disaster recovery. The third method is to distribute the server load among identical servers by using http redirect. The target server can use http redirect if it cannot accept more connections. This technique slows down the request processing since the request is sent back to the client along with another IP address, and then the client has to initiate another connection with the redirected server. This kind of load distribution is not transparent to the users. Another method to load balance the content servers is using IP anycasting, which is discussed in Section 2.2.

2.2 IP ANYCASTING

Server load can be balanced or distributed using IP anycast paradigm. Anycast is defined as the point-to-point flow of packets between a single client and the closest destination server identified by an anycast address. The idea behind anycast is that a client would like to send packets to any one of the several possible servers delivering a particular type of service or application. To accomplish this, a single anycast address is assigned to one or more servers contained within an anycast group. A client sends packets to a server in the anycast domain by placing the anycast address in the packet header. The network of routers will then attempt to deliver the packet to only one server with the matching anycast address. Anycasting schemes were mainly developed to balance the load among multiple servers.

IP Anycasting can be classified into network layer anycasting and application layer anycasting. In network layer anycasting, the routing protocols such as RIP and OSPF treat the anycast address as unicast addresses when exchanging routes. The client side routers store the anycast address paths as multiple routes to the same destination. When a client sends a HTTP request to an anycast server, router calculates the shortest path with the least cost route and transmits the packets to the closest server. The main advantage of this scheme is its simplicity and no modification requirement in the existing Internet infrastructure. However, this scheme has several issues. Anycast address space needs to be differentiated from unicast/multicast address space. Due to the stateless property of IP, the packet may be delivered to different servers each time. On the other hand, in application layer anycasting, IP anycasting is applied at application level rather than at network level. All members in a particular anycast group form an Anycast Domain Name (ADN) for identification. When a client wants a service, it sends request to local anycast resolver within its ADN. The resolver keeps metric information database and it chooses some IP addresses for the client. If more than one IP is chosen, a filtering mechanism applied at the client to determine the most wanted server IP. The client can adjust filter configuration. If the local resolver does not have the information of the client request, it will query with the authoritative resolver to get the information. Then the local resolver

calculates its own metric value and keeps it in the database. In this case, client sends an anycast query and gets back a unicast address of a server. So, there is no issue in TCP connectivity problem. Even though, this system provides load balancing, it is not scalable. It is very hard to keep the above infrastructure as the network grows. Local resolvers have to exchange information with authoritative resolvers frequently, causing congestion. Another problem routing proximity information cannot be accessed in the application layer, which is unlikely available and used in the network layer. Therefore, network layer anycasting is simpler when compared with application layer anycasting.

Network layer IP anycasting can be applied in the proposed Content Delivery Network (CDN) setup to achieve server load balancing. It identifies the proper server to achieve effective load balancing. It is a Decision Server (DS) or a router that makes the decision of which server to be selected, so this is a network-supported approach and the network path characteristics should also be considered to manage the network load at the same time.

As discussed above, there are mainly three issues in using IP anycasting to achieve server load balancing. Anycast address space has to be defined separately from the unicast address space to differentiate anycast addresses. The whole IPv4 address space has been already defined so there should be some other mechanisms to be implemented to identify anycast address space. Another issue is that, ICMP messages should not be sent with anycast addresses as destination addresses. It is because, then an ICMP message is generated with anycast address and it may be delivered to a different destination rather than the original destination.

Another critical issue arises due to IP anycasting is that TCP connectivity should be maintained between client and the server throughout the session. When a client sends a TCP_SYN packet with anycast address as destination address and it will be delivered to one of the servers in the anycast group. The server responds using anycast address as source address. When the client acknowledges it with anycast address, it could be delivered to a different server. There are several methods suggested to keep the TCP

connectivity of the client with the same server. RFC1546 proposed one method to ensure the TCP connectivity with one particular server, when a server receives a TCP connection request from a client with anycast address, it should respond with its unicast address, So that the client uses the unicast address as destination address in the subsequent packets [13, 16]. The main issue with this solution is that, it requires modification in the TCP semantics in both end points. R. Engel et al [7] suggested that the existing IP source route options could be used to bind the server's anycast address to the unicast address. In this scheme, the server responds through the unicast address interface using the anycast address as the source address. When the client sends subsequent packets, it still needs to keep the server unicast address in the IP source route option field. Performance could be degraded as intermediate routers must examine and process IP options field even it has nothing to do with the packet. Further, ICMP issue will arise as the anycast address is used as source address. M. Yamamoto et al [22] suggested using routers with active network technology, so that the active routers can recognize IP anycasting and replacing the anycast address with the selected server unicast address. In this method, the active routers should recalculate both IP header checksum and the TCP header checksum. Another solution to this problem is to develop an anycast to unicast address resolution or discovery mechanism that is performed separate from the TCP protocol. Client sends packets addressed to the anycast content server and in turn receives information about its unicast address. The HTTP protocol has a technique for redirecting client connections to a different host. In this case, the anycast server completes TCP connection setup with a client, recognizes it was dealing with an anycast address and then issues an HTTP redirect that instructs the client to establish a TCP connection with the unicast address of the server. One of these solutions can be employed in the proposed CDN architectures discussed in chapter 3.

While IP anycasting is used to achieve server load balancing, TE based request routing algorithm has to be used to optimize the distribution of load in the network. Next section covers the TE algorithms used in MPLS network environment.

2.3 MPLS TRAFFIC ENGINEERING

Figure 2.1: MPLS Network Architecture

The goal of Traffic Engineering (TE) in IP networks is to distribute or optimize network traffic and to avoid or minimize network congestion. Traditional IP networks provide best effort services over shortest path routing algorithm and fail to provide TE routing unless some modifications are done. Administrative policies, load distribution, Quality of Service (QoS) have become significant factors in Internet routing. Therefore, the path selection depends not only on the shortest path but also on a number of other factors. It is not a simple decision criterion based on number of hops or delay to a specified destination but the path selection should be based on multiple constraints such as available bandwidth, end to end delay etc. Constraint Based Routing (CBR) is a class of routing where the path selection is based on multiple constraints and service requirements in addition to reachability to the final destination. CBR is defined as the mechanism for computing a feasible network path for traffic trunks, based on a traffic description and a set of constraints. On the other hand, request routing enables the activity of directing a client request to a suitable content server. We use a TE friendly Request Routing (TFRR)

system in the proposed CDN architecture to route the request to a proper server to load balance the content servers and the network path load.

Traditional IP networks forward packets based on hop-by hop routing. MPLS networks forward packets based on label switching. It introduces a connection oriented forwarding plane based on label switching paradigm. Routers that support MPLS are called Label Switch Routers (LSR). The path setup is done by a signalling protocol and it is called Label Switched Path (LSP). An LSP originates at ingress LSR and terminates at egress LSR. Both of them are called Label Edge Routers (LER). The ingress LER (ILER) maps the incoming traffic onto LSPs based on Forward Equivalent Class (FEC). A FEC is a group of packets based on some common attributes such as destination address. All packets that match a given FEC will be sent on the LSP corresponding to the FEC. A shim header is defined to contain label information, between L2 and L3 headers. Routers forward the packets based on the label information in the shim header. When an LSR receives a packet it looks up the entry in the Label Information Base (LIB) and determines the outgoing label for the packet. The LIB in each node is created by exchanging labels for a given FEC. This is done by the label distribution protocols. Finally, the egress LSR removes the label from the packet and forwards it as a regular IP packet. In MPLS networks, explicit routes can be set up based on the metrics and constraints to facilitate TE.

CBR is used in traffic engineering for computing a feasible network path based on a traffic description and a set of constraints. Since route specification is determined in terms of constraints, it can dramatically reduce the amount of administrative explicit path configuration and manual intervention required to achieve TE objectives. Evolved from the concept QoS routing, CBR in MPLS network indicates to mapping a traffic trunk to a network path subject to a set of constraints. CBR process takes into consideration aggregation of traffic flows, whereas QoS routing deals with individual flows that are highly dynamic. Thus, the CBR process assumes traffic trunks that make large reservations and have long holding times. Also, CBR extends QoS routing by including

non- QoS constraints like administrative policies into path computation. Thus, CBR integrates two types of routing namely QoS based routing and policy based routing.

CBR processes can be divided according to their degree of integration within the network into off-line CBR and online CBR. In Offline CBR path computation takes place outside the network in a path computation server. This process takes a pre-empted static traffic as the input and computes optimal network paths for a given traffic demand. The main application of offline CBR is as a tool for network optimization planning. In contrast, online CBR is a routing process integrated with the network elements. It receives dynamic traffic demands as inputs, and based on dynamic updated network state, it computes a feasible path to satisfy the request. The two types of CBR have different characteristics each with distinct advantages and disadvantages. Offline CBR has no time restrictions to compute costly algorithms after taking into account detailed information about the network topology and traffic information. Thus, the computation is a global optimum. However, the main drawbacks of offline CBR are that a detailed traffic matrix has to be known in advance and there is no possibility of adapting to changes in the network state or accommodate new traffic demands. Online CBR process does not require a prior knowledge of network wide traffic demands, and it maintains update information of network state and copes with the network state changes. But the computation has to be done under strict operational requirements and has to be resilient to transient network conditions.

MPLS is a packet forwarding mechanism based on label switching, which was originally designed to support faster packet switching. In addition with constraint-based routing protocol and label distribution protocol it is now mainly used for traffic engineering. OSPF-TE is used as the enhanced version of the routing protocol for TE; while RSVP-TE and CR-LDP are the label distribution protocols used in MPLS networks. A path selection processor (PSE) is required to apply proper routing algorithm often termed as CBR, to optimize the operation of the network. The processor can be located either in the LER or in a dedicated server. Architecture of a typical TE-enabled LER is shown in Figure 3. The enhanced routing protocol and label distribution protocol perform TE

functions and interact with the PSE. The PSE takes the information provided by them as input for LSP setup. It obtains most updated topology details and resource usage from OSPF-TE and evaluates a LSP based on CBR. Then it communicates the LSP setup to RSVP-TE. The request includes explicit route, traffic class and amount of required resources.

Figure 2.2: Architecture of a TE enable LER node

2.4 BANDWIDTH GUARANTEED ALGORITHMS

There are a number of bandwidth guaranteed algorithms using available bandwidth information on the links, for example minimum hop algorithm (MHA), widest shortest path (WSP) algorithm, shortest widest path (SWP) algorithm, shortest distance path (SDP) and minimum interference routing algorithm (MIRA). MHA routes an incoming connection along the path that reaches the destination node using the minimum number of feasible links. This scheme, based on the Dijkstra's shortest path algorithm, is simple

and computationally efficient. However, using MHA can result in heavily loaded bottleneck links in the network, as it tends to overload some links while others remain underutilized. The cost of each link remains constant and independent of the current link load and therefore MHA tends to use the same paths until saturation is reached before switching to other paths with underutilized links. The WSP was proposed as an improvement of the MHA algorithm as it tries to load balance the network traffic. WSP selects the shortest feasible path. A path is feasible if its bottleneck bandwidth is equal or higher than the requested bandwidth. If more than one such path exists, WSP selects the path with the wider bottleneck bandwidth. However, WSP still has the same drawbacks as MHA since the path selection is performed among the shortest feasible paths until saturation before switching to other paths. On the other hand, SWP selects widest path among feasible paths. If there exists more than one such path, then the path with minimum hop count is chosen. In SDP, a feasible path with the shortest distance is selected. The distance of a path is defined as the sum of inverses of available bandwidths of its links.

The key idea in the MIRA scheme is to select a path that does not interfere too much with potential future LSP setup requests between other source–destination pairs. This algorithm uses network topology, residual capacity information, and the location of ingress–egress pairs. The algorithm computes maximum flow values for all ingress–egress pairs. Maximum flow value is an upper bound on the total amount of bandwidth that can be routed between source and destination. Then the critical links are identified as the links when an LSP is routed over those links, the max flow values of one or more ingress–egress pairs decrease. Once the critical links are identified they should be avoided on the routing computation to the maximum extent and thereafter, weights are computed for all ingress–egress pairs.

In the routing computation, initially all those links with residual bandwidth less than demand capacity are eliminated. Based on Dijkstra algorithm, a shortest path is determined on the reduced network. The demand units are routed along the chosen path and the residual capacities on each link are updated. MIRA gives superior performance

over MHA and WSP according to simulation tests. If all LSPs are long lived, new LSP requests are less likely to be rejected in MIRA compared to other algorithms. Further, MIRA provides very good rerouting performance on link failures in comparison with the other schemes.

Despite its more sophisticated functions, MIRA still has some limitations. MIRA discourages the use of critical links based only on the number of other source and destination pairs that could use them, without verifying if these source-destination pairs actually use the links. Hence, MIRA preserves the use of certain links that remain underutilized, thus causing a suboptimal use of the network. In its on-line implementation, MIRA sets the link weights almost in a static way according only to their level of criticality. In fact, the only event that can cause the redistribution of new weights is the saturation of some links, similar to the MHA. Another issue is that while choosing a path for an incoming request, MIRA does not take into account how the new call affects the future requests of the same ingress-egress pair.

2.5 REQUEST ROUTING ALGORITHMS

A typical Content Delivery Network (CDN) consists of several surrogate servers, a distribution system, a request routing system, and an accounting system. A request routing system enables the activity of directing a client request to a suitable surrogate (content) server. It consists of a set of network elements called request routers. They use dynamic information about network conditions and load on the surrogate servers to balance the load while directing the clients. The request routing system needs to maintain proximity metrics and server load metrics for this purpose. There are number of research works on possible solutions for CDN request routing problems. Routing solutions are based on DNS based request routing, Transport layer request routing, Application layer request routing, Content layer based request routing and NAT based request routing. In this section we present some previous works on request routing algorithms.

Z. Fu et al [8] proposed a combined path and server selection algorithm to route client to a proper server. The objectives of their design are to maximize the client success probability, and maximize the network and server utilization through load balancing techniques. The model incorporated three main factors: path, server, and end-to-end quality. The requirement in the selected path is the available bandwidth and server requirements, available capacities of CPU, memory buffer, disk bandwidth and network interface bandwidth. The relationship among the three parameters is that the end-to-end delay is the sum of path delay and server response time. The algorithm is aimed to provide a path that minimizes the bandwidth availability and the delay. A shortest path is computed subject to the availability and delay constraints in their algorithm and it is a restricted shortest path problem implemented using dynamic programming.

Even though, the MIRA was presented, the framework on which the algorithm to be used or could be effective could not be explained. Performance comparison over other request routing algorithms was not discussed. The computation complexity can be higher as it uses a restricted shortest path problem. As this algorithm is used for every client request, the processing delay should also be added to the total end to end delay, which will affect the combined path and server selection.

In scalable web request routing approach with MPLS, suggested by Arup Acharya et al. [1] an MPLS aware, intelligent client side proxy is combined with a high performance dispatcher. This MPLS based scheme takes advantage of mapping application layer information onto labels to enable high performance web switching. The MPLS enabled client proxy is responsible for mapping labels onto packets belonging to client connections. The label applied by the proxy is used at the dispatcher to choose which server should handle the request. The label-stacking feature of MPLS allows this inner label to be independent of any outer labels used to route the request through the network. The mapping of client connections to labels is communicated by the dispatcher to the proxy using a persistent control connection. Based on the mapping, the dispatcher can support a variety of functions. They are content based routing, client-server affinity, client specific service differentiation and server load balancing. However, this solution

has several deployment issues. This solution assumed wide deployment of MPLS in service provider network. The proxies at the edges of the network need to be modified to implement MPLS and support the protocol to communicate with dispatchers in server installations, which arises scalability issue.

R. Dragos et al [6] proposed another MPLS based open source web switching architecture for IP networks, to achieve web server load balancing. They focused on the load balancing of web servers in a cluster behind a dispatcher and preserve the functionality of MPLS to support CBR, client affinity, different classes of service and load balancing. In their architecture, a layer 4 filter placed at the ingress nodes classifies the SYN packets and label the packets with a dedicated SYN label marking at the beginning of a new session. The dispatcher is only responsible for deciding which the most lightly loaded server is, and then switch the incoming SYN label with the label associated for that server and forwards the packet to the server. The optimal server can be decided based on the processor load, the number of active connections, or the traffic generated through its network interface or even in a round-robin fashion. The server generates a usual reply, labels the packet and sends to the dispatcher. The packets originated from the server are relabeled at the dispatcher using an MPLS label stack. Another label is inserted to switch the packet along the MPLS cloud. The label added initially by the server remains valid along the path and is used to identify the server.

This scheme is implemented in a UNIX based environment and some performance studies are conducted. Round robin scheme is used in all test scenarios to achieve load balancing. However, test results on QoS parameters such as path load, delay are not measured. Further, this architecture only considers cluster of web servers. Our proposed architecture is suited not only for web clusters but also for geographically distributed servers. Path load is also taken into account when selecting the server. TCP connectivity is maintained based on Client Network Address Prefix (CNAPs) whereas this scheme uses label stacking which lengthens the label overhead.

Mirco Masa et al [12] proposed another request routing algorithm called Worst Case Exclusion (WSE). It excludes those surrogate servers that provide the client with low performance and prevent surrogate overloading. This algorithm could reduce user response time and effectively exploit the CDN capacity according to simulation results. WSE algorithm is based on three main concepts: the exclusion of surrogates that are too far according to a given criteria; the equalization of average response time, and the overloading prevention of the surrogate servers. The purpose of the algorithm is to minimize the user's average response time when they request the content using a CDN. It is composed of two phases. In the first phase, WSE excludes the surrogate with a value of latency higher than the estimated average response time. In the second phase, WSE calculates the probability coefficients of the remaining surrogates based on the link loads. If the link load values are below a threshold value, the algorithm behaves water-filling modality whereas link loads are above threshold; algorithm behaves according to overloading prevention modularity. However, determination of threshold value is arbitrary.

In the CDN delivery model, the request routing system processed the content requests that come from the different clusters, and it forwards them to the surrogates according to the request routing algorithm. The logical model is presented and the location where the RR system resided is not explained. The model used only exponential distribution of the content size and Poisson process of the request. It is a strong assumption and lack of precision in the estimation of one metric could compromise the benefits of the WSE algorithm.

2.6 CHAPTER SUMMARY

This chapter explains the background concepts and the previous related work on the related areas. Traditional methods of storing content are discussed at the start of the chapter. Then, IP anycasting paradigm, which is used for proper server selection is explained and the issues that arise are identified. Principles of Constraint based routing to achieve Traffic Engineering over MPLS is given in detail. At last but not least, some network architectures and request routing algorithms are presented.

3

Network Design Architecture

3.0 INTRODUCTION

A CDN provides hosting service to the content of web sites. Each web site is identified by a unique Fully Qualified Domain Name (FQDN) in the DNS server. In a typical CDN, the server loads are balanced for the server to accept and serve more number of client connections and improve the server response time. On the other hand, from client perspective the content access time should be minimal to gain the client satisfaction. The access time not only depends upon the server response time but also depends upon the latency of the path chosen to route the packets. Hence, the network resource management becomes critical in providing low latency paths for content delivery. It is equally important for operators to improve network utilization. In this thesis we propose an algorithm, called *TE Friendly Request Routing (TFRR)* algorithm, for the CDN architecture proposed in [10], to achieve load balancing among content servers, and at the same time, efficiently distribute network path load. In this chapter we discuss the CDN architecture, and the TFRR algorithm we discuss in Chapter 4.

In the CDN for TFRR [10], multiple content servers are located in different geographical locations. The content of a web site can be replicated on a group of servers that is identified by a single anycast address. The CDN is enabled with MPLS to perform traffic engineering. It employs TFRR to perform best available server and path selection. An explicit Label Switched Path (LSP) is chosen over the MPLS backbone such that the network traffic on the backbone is distributed to achieve traffic load balancing among the paths.

3.1 THE CDN ARCHITECTURE FOR TFRR

Figure 3.1: A CDN model to balance server load and network path load

Figure 3.1: The CDN architecture employing TFRR to balance server load and network path load.

The CDN architecture proposed in [10] is shown in Figure 3.1. Content servers are located across the geographical region. Content corresponding to a single web site is replicated on a group of servers, which is identified by an anycast address. Each server is configured with at least one unique unicast IP address and as many anycast addresses as the number of web sites hosted on the server. In the network both a single server or a server farm connected through a content switch called ‘dispatcher’ can be deployed. In the case of server farm, the anycast address is assigned to the dispatcher and the servers are configured with unicast IP addresses. The TFRR in this case selects the dispatcher (or the server farm), while the dispatcher selects a server within the farm. In the Figure, all servers replicate content and identified by the same anycast address. In this thesis we assume full content replication among all servers.

The Internet and the geographically distributed content servers are connected through a MPLS backbone. Label Edge Routers (LERs) that are installed at the edge where the backbone is connected to the Internet are called Ingress LERs (ILERs) as they receive the content requests from the Internet and forward them to a content server. Likewise, Egress LERs (ELERs) are the LERs located at the edge to connect the servers to the backbone. They receive content response from the servers and forward them to the ILERs. The LERs are interconnected through Label Switched Routers (LSRs) within the MPLS network. A Decision Server (DS) is centrally located much closer to the MPLS backbone, so that the LERs can establish low latency communication with the DS. The DS is designed with high processing power, memory and storage space. It performs several important functions of TFRR. It collects statistics on server load, traffic load, and information about network topology, resource availability, and policy details. Based on the up-to-date data, the DS applies TFRR algorithm to choose a proper server and then the best path to distribute the server load and traffic across the network. The DS selects the server and the Label Switched Path (LSP) and determines the resource to be reserved along the path. It, then, conveys its decision to the ingress and egress LERs. The ingress LER uses this information in binding the client network with the server, so that all packets from the client network can be directed towards the same server. The egress LER establishes the (reverse) LSP and reserves the resources along that LSP. The OSPF-TE protocol is used to flood the network state information across the network, which is discussed in Chapter 2. The DS learns about the updated network state through OSPF-TE. Thus, it can make path selections based on the traffic attributes, resource details and policy constraints.

Multiple CDNs based on the above design and independently managed by different operators can exist in the Internet, which serve the same content. In this case, although each network hosts the content of the same web site by replicating the content, they advertise different anycast addresses corresponding to the same web site, which identify their own group of servers. For example, the web site (www.abc.xyz) can be identified by n IP (anycast) addresses indicating that it is hosted in n CDNs. The global DNS system contains n records corresponding to each of those CDNs. When a request arrives for this

site, the DNS picks up one anycast address in a round robin fashion and sends that anycast address to the client, thus it performs first level request routing. Then, the client request goes to the CDN corresponding to the anycast address. Since anycast address is an address from the unicast address space, the global routing will forward the packet to the CDN that has advertised the anycast address. The resolution of the destination anycast address to a particular server location is performed by the TFRR in the CDN. In the remaining sections of this chapter we discuss the CDN design in detail.

3.2 DESIGN ISSUES

The main features of the CDN architecture, discussed in Section 3.1, are:

1. IP anycast address is used to identify a particular web site hosted in a particular CDN.
2. MPLS is used for traffic engineering in the CDN.
3. TFRR is designed to perform server and network path selections using the server and network load collected in real-time.

There are several issues we need to address before discussing the detailed design. First, what is the granularity of content delivery load that is used for making different server and path selections? In our design, we consider all the hosts generating content requests from a particular client network receive service from the same server over the same LSP. The clients can be identified by the common network prefix, called *Client Network Address Prefix (CNAP)*, of their IP addresses. Second, how to ensure that ILER always send the request to an anycast address coming from a CNAP to the same server? There are several options for this binding, which ranges from a dynamic situation where a decision server running TFRR is contacted every time a new TCP SYN packet is discovered, to a static situation where CNAP to server binding is downloaded to ILERs off-line. Different possibilities are discussed in the design choices in [10]. Third, since the content is sent from the server to the client in the reverse direction to the request coming from client to server, path load distribution should be achieved on the reverse path and not on the forward path. Therefore, the path is chosen and communicated to the

ELER, which receives the explicit route, sets up LSP along that route, and installs a Forward Equivalent Class (FEC) to bind the LSP with the Client Network Address Prefix (CNAP) and source port (SP). When this ELER receives data packets from the server it checks the CNAP and forwards the packets through the explicit LSP. Since the router installs FEC based on the CNAP, when the ELER receives content packets for another client with the same CNAP, it directly routes the packet through the same LSP.

Fourth, since anycast address is used for the server address, maintaining TCP connectivity between the selected server and the client needs further discussion. There are two important issues to be considered when an IP anycast address is used as the destination server address. First, all the packets originated from the client should be forwarded to the same server to maintain the client's session with the server. Second, when the data packets originated at the server destined to the client encounters an error in the network, inside the CDN or outside in the Internet, an ICMP message is generated from within the network, which must be returned to the original server. If the source address is an IP anycast address, the ICMP message may be delivered to another server rather than the source where the packet was originated. Thus, the network design and the forwarding algorithm should take these factors into consideration. There are several solutions available to deal with the client and server binding in handling ICMP error messages etc. We have presented a survey of the proposed mechanisms in Chapter 2. TCP connectivity between a client and the server can be maintained using either the source route option method described by R. Engel et al [7] or active network technology suggested by M. Yamamoto et al [22]. In the source route option method, the server unicast address is kept in the options field so that the routing could be done based on the unicast address. By using the active network's router option, packets originated at the server traverse from the server to the ILER with the server's unicast address as the source address. The ILER changes the unicast address to the anycast address in the source address field of the IP header by processing the router option and transmits the packets to the client. The client sends its packets to the anycast address with the same router option inserted in the packet. The ILER changes the destination address in the packet from the anycast to the unicast address by processing the router option, and then transmit the

packet to the server's unicast address inside the CDN. If an ICMP packet is generated in the Internet, then it arrives at the ILER through the Internet routing. The ILER needs to keep the binding between the source address and the server's unicast address to be able to forward the ICMP packet to the right server.

In another approach, a client sends the TCP_SYN packet to the anycast address, while the server responds with the unicast address. Afterwards, all subsequent packets from the client carry the server's unicast address as the destination address, hence the TCP connectivity is maintained and the ICMP packets are also delivered to the servers causing them. The TCP semantics at the client site needs to be modified to recognize the address change during the connection set up phase.

3.3 THE NETWORK DESIGN

Figure 3.2: The Design of CDN employing TFRR.

The DS has a priori knowledge of traffic demand from the client networks that it uses to predetermine the server and the path in the CDN and provision the LSP before the requests come from the client networks. The traffic demands of CNAPs can be estimated based on the historical data or based on the customer's Service Level Agreements (SLA).

The edge routers can do static LSP set up for the content delivery traffic destined to a particular CNAP. This provisioning scheme works for major content clients. The DS in this case can apply the TFRR algorithm, as discussed in Chapter 4, on their CNAPs offline. Once DS makes the selection for some CNAPs, it sends the binding information <Client Network Address Prefix; Server Unicast Address; LSP> to the relevant ELERs. All ILERs also receive <Client Network Address Prefix; Server Unicast Address> binding information from the DS for those CNAPs. In this scheme, clients always send their packets to the anycast address.

When a request arrives, the receiving ILER looks up the forwarding information table and finds the server unicast address and then forwards the request to that server through one of the available LSPs. When the sever responds, the ELER forwards the data packets through the given explicit LSP to achieve the network load balancing.

When the ILER receives an ICMP error message destined to an anycast address, it looks deep in the ICMP header, which carries the client address prefix information. The ILER uses the CNAP to identify the server unicast address to which it sends the ICMP error message through an available LSP. If the ICMP message is generated at an LSR within the MPLS backbone (including LERs), the label stack information is also carried in the ICMP header as the extensions to ICMP [2], so that based on the label information the LSR can return the ICMP message to the ELER, which then forwards the packet to the server. If the ELER is a dispatcher, it explores more information in the ICMP header and find out the client source address. Then it determines which server responds to the particular client. The dispatcher forwards the ICMP packet to that server.

The content traffic from the server always flow through the selected explicit paths to distribute the network traffic efficiently. When a new client from a client network whose LSP has already been provisioned sends a TCP_SYN request, the ILER looks up the matching CNAP entry in the forwarding table to determine the server address, and if it finds one then it forwards the request to the same server.

When a request arrives from a client network for which the LSP has not been provisioned in the CDN, hence the ILER looks up fails, then it forwards to a default server. The DS selects one or more default servers and paths for each ILER and conveys that decision to the each ILER. For the content delivery from default servers, default LSPs are also defined between every ELER, ILER pair. The default servers can be used in a round robin fashion to distribute the load. While default servers and LSPs are used for some CNAPs, the ILERs can send the information about those CNAPs to the DS. So that, the DS can monitor the CNAP load and provision LSPs for them [10].

The DS periodically monitors the server load and path load of the entire network. When it notices that one or more servers are heavily loaded, then the DS tries to change the server for those CNAPs which contribute excessive load to the over loaded server. The DS collects all the existing host specific TCP state information from the overloaded server for all the CNAPs and updates all ILERs with the host-specific IP address and server binding information. Hence, ILERs keep source address specific connections originated from the CNAP for the existing connections, which will be used in forwarding the traffic due to longest prefix match IP forwarding. The active TCP connection information can be obtained using *netstat* command.

The network path load can also be managed during the traffic flow even when the servers are not overloaded but the paths have become congested. Usually there are alternative paths available between and ELER and ILER, traffic flows can be easily migrated from one LSP to another by the request routing algorithm. By using a flow monitor or network traffic monitoring tool, server can monitor the throughput of all the traffic flow originated per each client address prefixes. It can send this throughput data to the DS. So the DS can use this data to compute path load on each LSP. For example, if one content server serves three client address prefixes with the throughput of 1Mbps, 2Mbps and 4Mbps and the first and third address prefixes use the same LSP to reach the ILER, then the path load is 5Mbps. Thus, the DS can keep track of LSP loads on all available LSPs. So that it can identify congested paths and then apply request routing algorithm to divert the traffic to alternate paths.

3.4 CHAPTER SUMMARY

We present an overview of the CDN design that employs TE friendly content delivery request routing in this chapter. The network design, functional behavior and the operational issues arise are fairly explored. In the design, all packets generated from the clients are sent to the anycast address and static server and path selection and their set up can be performed off-line, which reduces the client access time.

TE Friendly Request Routing Algorithm

4.0 INTRODUCTION

In Chapter 3 we discussed the CDN architecture to achieve server load balancing and network load distribution. The efficiency and the performance of load balancing and path load distribution primarily depend on the design of the request routing system. In conventional CDN request routing system forward requests to the least loaded servers in proximity to decrease the client response time. It achieves this goal by balancing the load among the servers. In this chapter we propose an algorithm for the request routing system implemented in the CDN discussed in Chapter 3, which performs both server and path selection in the network to achieve both server and traffic load balancing. We argue that by taking into consideration both loads, server and traffic on network paths, we can achieve the followings: (1) improvement in reducing client response time by not only using lightly loaded servers but also non-congested and lightly loaded paths, (2) better server and network resource utilization, and (3) in some cases increase the number of clients that can be served by not turning away clients that experience congestion inside the network. The request routing system uses a set of metrics in an attempt to direct users to content servers that can best serve the request. Such metrics include network proximity, bandwidth availability, server load and availability. And these metrics are based on the network measurements and feedback information given by the content servers.

4.1 PROBLEM DEFINITION

Consider a content distribution network with MPLS backbone that connects content servers to the Internet. There are three types of routers in the MPLS backbone network: ILERs connect the network to the Internet, ELERs connect the content servers to the CDN, and LSRs that connect ILERs and ELERs. Figure 4.1 shows an example CDN. One of the goals of our algorithm is to achieve load distribution among LSPs that carry content traffic from ELERs (content servers) to ILERs (client networks). We call these *reverse LSPs*. Request traffic is insignificant as compared to response (content) traffic, hence we do not consider selection of *forward LSPs* rather we assume that there is at least one LSP pre-established between every ILER and ELER pair. We define *forward LSP* as the LSP that carry request traffic from ILER to ELER. .

We assume the following notations to describe the proposed algorithm:

Figure 4.1: Network Diagram

Server Set: $S = \{S_1, S_2, S_3, \dots, S_l\}$, where each server S_j is directly connected to an ELER.

SC_j : Server Capacity of server S_j

Server Capacity represents the maximum number of TCP connections that the server can able to handle

SL_j : Server Load of server S_j

ELER Set: $E = \{E_1, E_2, E_3, \dots, E_m\}$

ILER Set: $I = \{I_1, I_2, I_3, \dots, I_n\}$

Path Set: $P = \{P_1, P_2, P_3, \dots, P_q\}$

Path Capacity : $PC = \{PC_1, PC_2, PC_3, \dots, PC_q\}$

Let the bottleneck bandwidth of a link L_i on the path P_α is BW_{P_α}

When a Path P_α is in between E_x and I_y ,

Links of the path P_α : $L_{P_\alpha} = \{L_1^\alpha, L_2^\alpha, \dots, L_g^\alpha\}$

Path load: $PL_\alpha = \sum BW_{L_i}^\alpha$

CNAP Set : $N = \{N_1, N_2, N_3, \dots, N_r\}$

Connection Load : $CL = \{cl_1, cl_2, cl_3, \dots, cl_r\}$

Traffic Load : $TL = \{tl_1, tl_2, tl_3, \dots, tl_r\}$

For every N_i there is (cl_i, tl_i) , where cl_i is the connection load and tl_i is the traffic load.

Find: (1) S_j , and (2) P_k

Subject to constraints

$$SL_j + cl_i \leq SC_j \quad \dots\dots\dots (1)$$

$$PL_k + tl_i \leq PC_k \quad \dots\dots\dots (2)$$

Objectives

1. Server load balancing

$$SL_j - SL_x \leq \epsilon, \forall S_x \in S \text{ and } x \neq j$$

2. Path load balancing

$PL_k - PL_y \leq \varepsilon, \forall P_y \in P, y \neq k$, and P_k and P_y are between the same ELER and ILER.

4.2 SERVER LOAD METRICS

The computation of the metric depends on the collection of updated server load information and network path details. How much information is required and how frequent the information is updated are the main issues in collecting load information. The server load measurement can be based on several parameters such as number of TCP connections served, the disk load, available CPU capacity or CPU idle status, load average, memory usage, and network interface card bandwidth. These parameters can be individually measured, and a composite weight index can be obtained by combining all the necessary parameters linearly. Otherwise, the load can be represented based on a single measure such as number of TCP connections established.

In the CDN design considered in this thesis, as discussed in Section 3, the request routing forms FEC based on the Client Network Address Prefix (CNAP). The *server load* is generally measured as the number of connections being served. We define *CNAP load* as the expected number of client requests from a network identified by the CNAP. The *server capacity* of a server is defined as the maximum number of connections the server is able to handle at a particular instance. For the algorithm proposed in this chapter, we consider the aggregation of CNAP load generated by a server to be its server load, which is given by:

$$SL_j = \sum_{all_k} P_k$$

// SL_j is server load for server j .
// P_k is the number of connections served in each CNAPs.
// k is the number of CNAPs

CNAP load can be estimated from the historical or contractual data, which is termed as expected *CNAP load*. The prefix length of the CNAP determines the maximum number

of clients that can launch requests from a client network. The CNAP load can also be estimated based on the past traffic patterns of the same network prefix. This information can be retrieved from the customer who uses the CNAP when signing the Service Level Agreements (SLAs). The existing server load (SL) and the expected CNAP load are subtracted from the server capacity (S) to get the Residual Server Capacity (R).

Hence,

$$R_j = SC_j - [SL_j + \text{CNAP load}]$$

// R_j is the residual capacity of server j.

// SC_j is the server capacity for server j

// SL_j is server load for server j.

// CNAP load is the expected load from the arriving CNAP.

The following feasibility test is performed for each server, and a set of feasible server is computed.

For EACH SERVER

IF $R_j > 0$

FEASIBLE; ADD to FEASIBLE SERVER SET

ELSE

INFEASIBLE;

LOOP

FEASIBLE SERVER SET: $\{FS_j \mid R_j > 0; j = 1, 2, 3, \dots, n\}$

If the feasible server set contains more than one server, then the server is selected based on the path selection.

4.3 PATH LOAD METRICS

For path selection, the following three factors are considered:

- [1] Demand interference
- [2] Criticality
- [3] Link utilization

The measure of *demand interference* shows whether a link causes interference to other paths between any source-destination (S, D) pair, which is between any ELER, ILER pair. The notion of interference is first defined in the MIRA [11] as a path that does not interfere too much with potential future requests (demands) between other source destination pairs". W. Sa-Ngiamsak et al [17] developed an algorithm based on this measure. The demand interference weight (DW) for each link is computed based on the following formula

$$DW_i = \frac{\text{Possible_demand_path_on_Link}_i}{\text{Total_possible_demand_path}} \text{ for Link}_i$$

// DW_i is the demand weight for Link i .

The numerator is the number of paths between any ELER, ILER pair that use link i , and the denominator refers to the total number of paths available in the network between all (ELER, ILER) pairs. So the maximum value of DW is 1, which indicates that the link is almost used by all the paths in the network. Hence, the algorithm should select the path whose links have the low DW values. This demand interference weight is a static measure, which is calculated once. It is recomputed as a result of change in topology.

The *criticality* of a link measures how critical the link is to a path selection for a (ELER, ILER) pair. For example, consider the case where only one path is available between a given (ELER, ILER) pair, while the links along the path may have lower DWs. If the LSPs between other (ELER, ILER) pair use one or more links along this path, no traffic

could be sent between the given (ELER, ILER) pair, since the only path is blocked and used by the other pairs.

The critical weight (CW) is computed for each link i as follows:

FOR EACH link _{i}

 LIST the (ELER _{i} , ILER _{j}) pairs which uses link _{i} (n)

 FOR EACH (ELER _{i} , ILER _{j})

 COUNT the number of ALL POSSIBLE PATHS (n_k)

 CRITICAL FACTOR (C_k) = $1 / n_k$

 LOOP

$CW_i = \text{MAX} [C_1, C_2, \dots, C_n]$

LOOP

The value of CW is between 0 and 1, and the maximum value of 1 indicates that the link is almost critical and trying to route some traffic will block other (ELER, ILER) pairs. The algorithm avoids selecting those paths whose constituent links have high CW value. Like demand weight, it is also a static measure, which changes if the topology changes.

The third factor, *utilization*, is included to achieve the distribution of the network traffic load. The weight is based on the link utilization. The utilization can be measured from either the actual load on the link or the expected load on each link based on the expected traffic of each CNAP.

If a path is chosen for one CNAP, all new clients from the same CNAP will be routed through the same LSP to the same server. So that, it is better to use the expected link utilization of a link which carries some traffic from some CNAPs, and reserved bandwidth on number of expected connections for each CNAP. However, keeping more reservations will lead to lower overall network load distribution, and some new requests could have a chance to be rejected. On the other hand, selecting the path based on actual link load could be under estimate. When more number of requests arrive from existing CNAP clients, the network will experience congestion and hence data loss will occur.

The utilization weight for each link is computed as follows

$$UW_i = \frac{\text{Utilized_bandwidth_of_Link}_i}{\text{Total_bandwidth_of_Link}_i} \text{ for all Link}_i$$

// UW_i is the prefix weight for Link i

The UW has values between 0 and 1. A value close to 1 indicates that the link is either heavily loaded or could be used by more future requests of existing CNAPs. A value close to zero means the link is not congested and a potential for a new path selection.

Altogether, three different weights are computed namely *demand interference weight* (DW), *criticality weight* (CW) and *utilization weight* (UW) to reflect the interference to other (ELER, ILER) pairs, criticality or blocking of other (ELER, ILER) pairs, and traffic load of a link respectively.

4.4 SELECTION PROCESS

The algorithm performs server selection and path selection in two distinct steps. Initially, it computes a feasible server set based on server residual capacities as explained in Section 4.2. An average weight (AW) is computed using the three path load metrics as explained in Section 4.3. The three weights, DW, CW, and UW are multiplied by parameters a , b , and c respectively, and then added to get the average weight. The parameters a , b , and c control the influence of different factors on the average w .

For each Link i

$$AW_i = \alpha.DW_i + \beta.CW_i + \gamma.UW_i \quad i = 1, 2, 3, \dots, n$$

Where α , β , and γ are constants with default values to be 1. That means all three weights have equal effect on the final average weight. Initially, when there is not much utilization on the links, the utilization weight will be lower and hence its effect on the final weight is minimal. Utilization weight may keep a higher coefficient than the other two weights to

make equal effect on the average weight. Criticality factor may also be given higher priority than demand interference. The average weight cannot be used in the initial server-path selection, which is explained at the end of this chapter.

The average weight of each link is computed, which is used as a metric in the shortest path computation during the path selection process.

For a feasible server (FS_j) in the feasible server set, the $ELER_j$ is the router that directly connects the server. For the path selection, the algorithm first computes path weigh, PW , of all the paths originating from the $ELER_j$. The PW of a path is simply the sum of AW of all the links along the path.

```

FOR EACH  $FS_j$ 
    GET PATH (P) s originating from  $ELER_j$ 
    FOR EACH  $P_i$ 
         $PW_i = \sum AW_k$ 
        //  $PW_i$  – Weight of the  $i^{th}$  LSP of  $ELER_j$ 
        //  $AW_k$  is the weight of the link  $k$ ,  $k = 1, 2, \dots, n$ 
    LOOP

```

The minimum PW value of all the paths originating at $ELER_j$ is taken as minimum path weight ($MiPW$). The maximum path weight ($MaPW$) is computed by multiplying $MiPW$ with the load sharing ratio f as follows.

Min. Path Weight : $MiPW = \text{MIN } [PW_i, i = 1, 2, 3, \dots, n]$

Max. Path Weight : $MaPW = MiPW * f$

Then only the feasible paths (FP) having path weights between $MiPW$ and $MaPW$ are chosen.

$$FP_i = \{P_i \mid MiPW \leq PW_i \leq MaPW\}$$

For each feasible server (FS_j) as computed in Section 4.2, dividing the residual capacity (R_j) by the server capacity (SC_j) yields the Residual Index (RI_j)

$$R_j = SC_j - [SL_j + \text{CNAP load}]$$

$$RI_j = R_j / SC_j = 1 - [SL_j + \text{CNAP load}] / SC_j$$

Finally, a path in the feasible path set, connecting to a server in the feasible server set which has the Maximum RI, will be chosen. The Server with the maximum RI will be selected.

Selection:

$$[\text{Server, Path}] (S^*, P^*) = \{FS_j \mid \text{MAX}(RI_j) \mid \{S^*\} \subset \{FS\}; P_i \mid P^* \subset \{FP\}\}$$

If the feasible server with maximum RI has more than one path, then the path with a minimum PW will be chosen and a LSP will be created along the chosen path.

Selection of possible paths from each ELER that directly connects to one or more feasible servers can be restricted by only considering the available paths between the ELER and the ILER that forwards the CNAP request. So the choice of path selection becomes narrower. In general, paths that connect ELER to all ILERs are considered.

Following are the possible benefits of this algorithm:

- Provides better server response time and able to accommodate more future requests since the server with the maximum residual capacity is chosen.
- Achieves server load balancing, since a feasible server is chosen based on the server load measurement.

- The path selection does not affect future path selection for the requests from new CNAPs as well as the existing CNAPs as the least interfered and least critical links are selected.
- Obtains network load distribution as the network path selection process choosing the path based on the utilization or current traffic on each link. So benefits of traffic engineering could be achieved.

4.5 THE ALGORITHM

The TE friendly Request Routing (TFRR) algorithm is presented here.

Periodic Feasible Server Set Selection

```

New CNAP request arrives
DO WHILE true
    // Perform feasible server set selection (ELER set)

     $SL_j = \sum_{all\_k} P_k$  for all Serverj
        //  $SL_j$  is server load for server j
        //  $P_k$  is the number of connections served in each CNAPs
        // k is the number of CNAPs

     $R_j = SC_j - [SL_j + CNAPload]$ 
        //  $R_j$  is the residual capacity of server j
        //  $SC_j$  is the server capacity of server j
        //  $SL_j$  is server load of server j
        // CNAP load is the expected load of arriving CNAP

    FOR EACH server
        IF  $R_j > 0$  THEN
            Add this server j into Temporary Feasible Server Set.
        END IF
    LOOP

    Update Temporary Feasible Server Set to Feasible Server Set.

    Sleep for n seconds.
LOOP

```

Demand Interference Weight Computation

Listen for updates from routing protocol.

When route update is received:

// Perform demand weight calculation.

List all combination of all possible paths.

Route all paths with request bandwidth of one unit.

$$DW_i = \frac{\text{Possible_demand_path_on_Link}_i}{\text{Total_possible_demand_path}} \text{ for all Link}_i$$

// DW_i is the demand weight for Link i .

Store DW_i for all i into Demand Weight Table.

Critical Weight Computation

Listen for updates from routing protocol.

When route update is received:

// Perform demand weight calculation.

FOR EACH link _{i}

List the (ELER _{i} , ILER _{j}) pairs which uses link _{i} (N)

FOR EACH (ELER _{i} , ILER _{j})

COUNT the number of ALL POSSIBLE PATHS (N_k)

$$C_k = 1 / N_k$$

// C_k is the Critical factor of k^{th} pair

// N_k is the number of possible paths between k^{th} pair

LOOP

$$CW_i = \text{MAX}[C_1, C_2, C_3, \dots, C_k]$$

// CW_i is the critical weight for Link i .

Store CW_i for all i into Critical Weight Table.

Sleep for n seconds

LOOP

Periodic Average Weight Calculation

```
DO WHILE true
    // Perform Utilization weight calculation

    
$$UW_i = \frac{\text{Utilized\_bandwidth\_of\_Link}_i}{\text{Total\_bandwidth\_of\_Link}_i}$$
 for all  $\text{Link}_i$ 
    //  $UW_i$  is the utilization weight for Link i

    
$$AW_i = a.DW_i + b.CW_i + c.UW_i$$

    //  $AW_i$  is the average weight for Link i.
    //  $DW_i$  value is retrieved from Demand Weight Table
    //  $CW_i$  value is retrieved from Critical Weight Table
    a,b,c are constants and the default values are 1.

    Store  $AW_i$  for all  $i$  into Average Weight Table.

    Sleep for n seconds
LOOP
```

LSP Weight Setting

Listen for updates for Average Weight Table, and Feasible Server Set.
When an update is received, perform Path table lookup.

```
FOR EACH LSP entry
    IF LSP points to server which exists in Feasible Server Set THEN
        Set  $PW_i$  to  $\sum AW_k$  for all  $\text{link}_k$  on the path.
        Store  $PW_i$  for all  $i$  into Path Table.
    ELSE
        Set PW to a maximum value.
    ENDIF

LOOP
```

Periodic Feasible Path set Calculation

// Perform feasible path set selection

Path Table is a set of paths that have available bandwidth greater than required bandwidth.

$$MiPW = MIN[PW_i]$$

// PW_i is the path weight of LSP_i

// $MiPW$ is the Minimum Path Weight

$$MaPW = MiPW * f$$

// $MaPW$ is the Maximum Path Weight

// f is the load sharing ratio

Lookup PATH Table

FOR EACH PATH entry in Path Table

IF $MiPW \leq PW_i \leq MaPW$ THEN

Add this PATH into Temporary Feasible Path Set.

END IF

LOOP

Update Temporary Feasible Path Set to Feasible Path Set.

Sleep for n seconds.

LOOP

Best Server-LSP Selection

When an incoming packet is received, perform lookup in both Feasible Path table and Feasible Server Set.

Send the packet to choose the server with maximum Residual Index (RI) through the LSP with minimum weight.

$$RI_j = R_j / SC_j = 1 - [SL_j + CNAPload] / SC_j$$

for all $Server_j$ in Feasible Server Set

// RI_j is the residual index of server j

// R_j is the residual capacity of server j

// SC_j is the server capacity of server j

// SL_j is the server load of server j

// CNAP load is the expected load of arriving CNAP

Initial Server-Path Selection

When a request is received first time

Server Selection

Server with the Maximum number of available paths is chosen

Then

Path Selection

For EACH PATH in $(ELER_s, ILER_i)$ where $ELER_s$ is the set of ELERs connecting the chosen server

$$PCW_i = MAX[CW_1, CW_2, CW_3, \dots, CW_k]$$

// PCW is path critical weight of the i^{th} path with k number of links

Path with the minimum path critical weight is chosen

If there are multiple paths exist with minimum path critical weight

THEN

Shortest path based on average weight is chosen.

4.6 CHAPTER SUMMARY

This chapter proposes the TFRR algorithm to route the requests to a least loaded server over a less interfered, least loaded path. A metric is computed based on the demand weight, criticality weight and the prefix load weight and then the shortest path first algorithm is applied based on the average metric and based on the residual index of feasible servers and within the feasible path list, a server and a path will be selected. The algorithm in detail is presented.

5

Simulation & Results

5.0 INTRODUCTION

In this chapter we present the simulation set up and then discuss the results. An important component of the simulation set up is realistic workload models, which is critical for proper capacity planning of content delivery infrastructures. Workload characterization is fundamental to the synthesis of realistic workloads. We discuss different workload models we used in our simulation. We developed a simple CDN simulation model in ns-2 to show the feasibility of our approach. Then, we present the results of more detailed simulation in OPNET.

5.1 WORKLOAD CHARACTERIZATION

Many studies are focused on hierarchical workload model when the researchers want to develop workload model for video streaming [20]. A hierarchical workload model basically consists 4 layers: Client layer, Session layer, Transfer layer and Traffic layer.

Client layer focuses on characterizing the client population. A unique ID such as IP address identifies a client. Our TE friendly architecture looks on the CNAP rather than a single client with an IP address. So, the client layer can be identified using CNAP. This also reduces the number of client entries in the system. Other characteristics include number of clients accessing the content and the client inter-arrival times. Usually, client arrival process is a Poisson process. Hence the inter-arrival time follows an exponential distribution. Active clients of a live content access can follow exponential or Zipf distribution.

The next layer in the hierarchy is session layer. A client can have multiple sessions on a single request. For simplicity, it could be assumed that there is only one session per request. This is a valid assumption and that reduces much complexity in the session layer. During the session, the client can do multiple transfers. Normally when a client arrives, he/she starts the session immediately and does the first transfer at that time. Then they take an OFF time to view the video and then do transfers several times in the same way. Here, the transfer layer characteristics come and play. Number of transfers in a single session follows a Pareto distribution. Each transfer consumes a time called the transfer length and it follows lognormal distribution. The time between the arrivals of transfers also follows a lognormal distribution. At the lower level, the traffic layer characteristics is defined to the traffic attributes such as bandwidth usage, packet size and packet inter-arrival times. This focuses on the network and transport layers.

Table 5.1 summarizes the variables of the workload model of a streaming media with the parameters based on the survey conducted by E. Veloso et al [20]

Workload Variable	Distribution	Parameters (Std.)
Client Arrival Rate	Poisson	$\lambda = f(t)$
Client Interest	Zipf	$\alpha = 0.470; \beta = 0.001$
Transfers per session	Pareto	$a = 1.43; b = 0.62$
Transfer length	Lognormal	$\mu = 4.29; \sigma = 1.28$
Transfer inter-arrival	Lognormal	$\mu = 4.93; \sigma = 1.26$

Table 5.1: Workload parameters

We followed the streaming media workload model for our content model used in the client profile. This model is applied in our detailed simulation using OPNET tool.

5.2 NETWORK SIMULATORS

Simulation is essential to validate the research objectives. Simulation modeling becomes increasingly popular for network performance analysis. There are two types of network simulation methods. One is the analytical modeling and the other one is computer simulation. The former is based on mathematical analysis, which usually makes some assumptions to simplify the model. The problem of mathematical analysis is that due its simplicity cannot simulate the dynamic nature of a network. Therefore, study of a complex network system always requires a discrete-event network simulator, which can compute the time that is associated with real events in realistic manner.

Simulation software is a valuable tool especially for today's network with complex network architectures. Traffic engineering research often needs network simulators in order to facilitate development and analysis of new traffic management and control schemes. Network simulators can be either open source or commercial. The open source simulators are free and usually developed and maintained by academic institutions, and they are easy to modify and add new features. Since they are developed by different people with no support from a single company, designing a detailed simulation mode becomes a daunting task. The well-known open source simulator is NS-2 [14]. On the other hand, the commercial simulators are supposed to deliver valid network models. One of the most popular software is the OPNET Modeler developed by OPNET.

5.3 SIMULATION USING NS-2

We simulated a simple CDN to evaluate the benefit of a request routing system that distributes LSP loads proportional to the demand as opposed to the distribution of load in a round robin fashion in case of DNS-based request routing. The proportional load distribution is one of many schemes that can be implemented using the proposed TFRR system. We first describe the simulation set up and then discuss the results in this section.

Figure 5.1: Simulation Topology

5.3.1 SIMULATION SET UP

We used NS-2 [14] to simulate the proposed CDN using the topology shown in Figure 4.1. We consider four identical content servers, nodes 7-10 in terms of their capacity and transmission rates, which are connected through multiple network paths with the client node 0. The nodes 1 to 6 form a simple CDN where all the requests enter the CDN through a single ILER at node 1, which performs the path and server selection. The ELER at nodes 3-6 connect the servers 7-10 respectively. The link 1-2 is configured with enough bandwidth to accommodate the traffic across all other paths so that it doesn't become a bottleneck link. The links 2-3 and 2-4 are configured with 64kbps, whereas 2-5 and 2-6 with 128kbps.

In a round robin server selection scheme each server gets the same number of requests, whereas in the proportional scheme servers 9 and 10 receive almost twice the number of requests server 7 and 8 receive. We generated 30 identical http requests at the same time each requesting 1KB of content. In case of round robin nodes 7 and 8 serve 8 requests each, whereas nodes 9 and 10 serve 7 requests each. In case of proportional load

distribution server selection scheme nodes 7 and 8 receive 5 requests each, whereas nodes 9 and 10 receive 10 requests each.

5.3.2 SIMULATION RESULTS

We used two performance measures – the response time and the path utilization. Response time of a client request is defined as the time when the client sends the content request to the time it receives the last packet of the content. The primary objective of all CDNs is to achieve low response time, which is the main client’s performance indicator. The secondary objective of the proposed CDN is to achieve fair distribution of traffic loads across LSPs.

Figure 5.2: Response time of web requests for (a) proportional load distribution and (b) round robin load distribution

Figure 5.2 shows the average response time computed at each server over all the requests served by that server for both round robin and proportional schemes. It is evident that the response time of the clients served by the servers connected with the high speed links (e.g. servers 9 and 10) is almost half as low as the response time of the clients served by the servers (7 and 8) connected with the low speed links. The difference in response time is almost in proportion to the difference in the link speed. In case of proportional load distribution the overall average time is decreased and become almost the same for all the servers. However, the response time of the clients served by servers 9 and 10 have increased slightly.

Figure 5.3 shows the utilization of links connecting each server. It is evident that the loads on those links are fairly equally distributed in the proportional scheme. In contrast, round robin request routing shows uneven load distribution across those links, as the slow links receive as much traffic as the high-speed links.

Figure 5.3: Load distribution under each scheme

The simulation results show that the proposed TFRR scheme has a potential of achieving the same objective as of DNS-based or any other request routing system. Further, it also tends to achieve its secondary objective of distributing the traffic load inside the network.

This simulation setup doesn't use the realistic workload model that is described in section 5.1. We developed a detailed simulation model in OPNET that uses the above workload models.

5.4 SIMULATION USING OPNET

OPNET (Optimized Network Engineering Tool) [15] provides a comprehensive simulation environment for the specification, simulation and performance analysis of communication networks. OPNET modeler is an object oriented simulation tool that provides a visual simulation environment for network modeling. OPNET is structured into number of hierarchical modeling layers similar to communication protocols. The advantages of using OPNET include the following: (1) it is more versatile to model

various network protocols under wide range of operating conditions, (2) it provides users with powerful modeling, simulation, and data analysis facilities and (3) it is more user-friendly.

5.4.1 THE WORKLOAD MODEL IN OPNET

A realistic workload model is developed in OPENT according to the parameters discussed in 5.1. This model is applied in the simulation scenario developed in the OPNET. We chose web browsing application to develop workload model. The application *config object* is configured with the HTTP application. Servers are configured to support the HTTP application.

The profile is configured in accordance with the realistic workload model. The configuration is shown in the Figure 5.4. The profile *web traffic* is configured with the application *WEB*. The application duration follows lognormal distribution with the parameters shown. The time between the applications also follows lognormal distribution and the numbers of repetitions are based on Pareto distribution. The application lasts for 250s. The profile is repeated based on exponential distribution among the clients. The number of repetitions or the concurrent users is based on Zipf distribution. One client request is independent of another client so that, the profile repetition is concurrent. This profile setup is applied to all the clients in the network scenario.

Figure 5.4: The profile configuration.

5.4.2 CDN SCENARIOS

Conventional CDN: Conventional IP routing and Round Robin Server Load Balancing [Conventional]

Figure 5.5: Simulation Network Topology – Conventional CDN

The basic network topology for simulation is shown above. Three identical content servers with same capacity are placed in dispersed locations. They are running web content application. All of them are connected to the IP backbone through high-end routers. All routers are identical routers. The network runs OSPF as the main routing protocol. All links have equal costs. Ten different subnets are created in the client network side as shown in Figure 5.6. The gateway connecting various subnets is connected to the ingress routers I1 and I2.

Figure 5.6: Client subnets

A typical client setup within a subnet is shown in figure 5.7.

Figure 5.7: Client Network

Server and client workstations are connected to a Hi-speed switch using standard 100BaseT link. Each client network works independently. In this model, servers are chosen by the clients in a round robin manner, as chosen in the DNS based server selection. Clients request content application with three different profiles. Profiles were created according to the standard workload models. We run simulation for a half an hour with the following setup.

- Clients send requests for web content application according to the profile configuration setup.
- The request will be served by one of the three content servers in round robin fashion.
- The server responses are routed along the shortest paths based on the link weights set by the traditional OSPF protocol.
- The TCP load on each server and the traffic sent on each link were monitored.

Proposed CDN: TE Friendly Request Routing and Round Robin Server Selection [TFRR + RR]

We developed this model to test the proposed TE friendly request routing algorithm combined with the round robin server selection. MPLS backbone network is used to facilitate TE and explicit routing. The network model for this scenario is shown in Figure 5.8

Client subnets and content servers are connected through MPLS backbone, which provides fast switching of labels and facilitate Traffic Engineering capabilities. Client subnets are connected to the MPLS network through ILERs namely I1 and I2; whereas, servers are connected to ELERs namely E1, E2 and E3. Reverse LSPs are formed between ELERs and ILERs to carry the server response traffic. Round robin scheme is used for server selection as in conventional CDN.

Figure 5.8: Simulation Network Topology – Proposed CDN

The traffic load matrix is created by measuring the traffic from each source to destination in conventional CDN. The traffic load to each client is measured and shown in Appendix 1. The traffic load matrix is shown in Table 5.2

Source / Client (kbps)	Server1	Server2	Server3
Subnet 0	89.4	251.6	76.8
Subnet 1	101.4	89.0	203.6

Table 5.2: Traffic load matrix

The path selection in TFRR algorithm is based on of the average of demand interference weight, critical weight and utilization weight. The first two measures are static measures that are computed as follows.

The MPLS network diagram is shown below. ILERs and ELERs act only as starting nodes or finishing nodes and not as intermediary nodes. In the network there are 14 unique paths connecting every pair of ILER and ELER.

Figure 5.9: MPLS network diagram

Available Paths

P1. E1 – L2 – L1 – I1	P8. E2 – L3 – L2 – L1 – I1
P2. E1 – L2 – L3 – L1 – I1	P9. E2 – L3 – I2
P3. E1 – L2 – L1 – L3 – I2	P10. E2 – L2 – L1 – L3 – I2
P4. E1 – L2 – L3 – I2	P11. E2 – L2 – L3 – I2
P5. E2 – L2 – L1 – I1	P12. E3 – L3 – L2 – L1 – I1
P6. E2 – L2 – L3 – L1 – I1	P13. E3 – L3 – L1 – I1
P7. E2 – L3 – L1 – I1	P14. E3 – L3 – I2

Link	Demand	Demand Interference Weight
E1 – L2	4	4/14 = 0.286
E2 – L2	4	4/14 = 0.286
E2 – L3	3	3/14 = 0.214
E3 – L3	3	3/14 = 0.214
L2 – L1	6	6/14 = 0.428
L3 – L2	6	6/14 = 0.428
L3 – L1	6	6/14 = 0.428
L1 – I1	8	8/14 = 0.571
L3 – I2	6	6/14 = 0.428

Table 5.3: Demand weight table

Criticality Analysis

(ELER – ILER) pairs and available paths

Available paths (E1 – I1) = 2; Available paths (E2 – I1) = 4; Available paths (E3 – I1) = 2;

Available paths (E1 – I2) = 2; Available paths (E2 – I2) = 3; Available paths (E3 – I2) = 1;

Link	Critical factors (CF)	Critical Weight = Max (CFi)
E1 – L2	0.50, 0.50	0.50
E2 – L2	0.25, 0.33	0.33
E2 – L3	0.25, 0.33	0.33
E3 – L3	0.50, 1.00	1.00
L2 – L1	0.50, 0.25, 0.50, 0.50, 0.33	0.50
L3 – L2	0.50, 0.25, 0.50, 0.50, 0.33	0.50
L3 – L1	0.50, 0.25, 0.50, 0.50, 0.33	0.50
L1 – I1	0.50, 0.25, 0.50	0.50
L3 – I2	0.50, 0.33, 1.00	1.00

Table 5.4: Critical weight table

Initially, the link utilization is zero for all the links, as there is no traffic on the network. LSPs are assigned for each source and destination according to their traffic load in the following order.

1. Server2 – Subnet0
2. Server3 – Subnet1
3. Server1 – Subnet1
4. Server1 – Subnet0
5. Server2 – Subnet1
6. Server3 – Subnet0

Routing for the traffic of first source-destination pair is based on the initial server-path selection. In this model, server is already chosen as to route the traffic originating from Server2. There are seven alternative paths available to reach Subnet0 from Server2. Paths connecting E2 – I2 have a critical weight of 1 while paths connecting E2 – I1 have a

critical weight of 0.5. There are four alternate paths available and the shortest path is chosen based on the average weight. The selection detail is shown in Appendix1.

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.252	0.799
E3 – L3	0.214	1.000	0.000	1.214
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.252	1.180
L1 – I1	0.571	0.500	0.252	1.323
L3 – I2	0.428	1.000	0.000	1.428

Table 5.5: Average weight table

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.323		3.037
E1 – L2 – L3 – L1 – I1	0.786	0.928	1.180	1.323	4.216
E1 – L2 – L1 – L3 – I2	0.786	0.928	1.180	1.428	4.322
E1 – L2 – L3 – I2	0.786	0.928	1.428		3.142
E2 – L2 – L1 – I1	0.619	0.928	1.323		2.870
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.180	1.323	4.049
E2 – L3 – L1 – I1	0.799	1.180	1.323		3.301
E2 – L3 – L2 – L1 – I1	0.799	0.928	0.928	1.323	3.977
E2 – L3 – I2	0.799	1.428			2.227
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.180	1.428	4.155
E2 – L2 – L3 – I2	0.619	0.928	1.428		2.975
E3 – L3 – L2 – L1 – I1	1.214	0.928	0.928	1.323	4.393
E3 – L3 – L1 – I1	1.214	1.180	1.323		3.716
E3 – L3 – I2	1.214	1.428			2.642

Table 5.6: Path weight table

Then, the path between Server3 and Subnet1 is chosen using the shortest path algorithm. Table 5.6 shows as E3 – L3 – I2 be the shortest path between Server3 and Subnet1. An LSP is created along this path and the traffic between Server3 and Subnet1 are routed along this LSP. The utilization weights, average link weights, and path weights are

updated before routing the next traffic. A route will be chosen based on the updated path table. This process is repeated until the paths are chosen for the entire source – destination pairs. The link weights and path tables are shown in Appendix 2.

The paths selected for each source – destination pairs are given in Table 5.7. Separate LSPs are created along each of those paths. Subnet address of each destination subnet is used to form FEC for the respective path.

Source – Destination	Server	LSP	Path
Server2 – Subnet0	Server2	LSP1	E2 – L3 – L1 – I1
Server3 – Subnet1	Server3	LSP2	E3 – L3 – I2
Server1 – Subnet1	Server1	LSP3	E1 – L2 – L3 – I2
Server1 – Subnet0	Server1	LSP4	E1 – L2 – L1 – I1
Server2 – Subnet1	Server2	LSP5	E2 – L3 – I2
Server3 – Subnet0	Server3	LSP6	E3 – L3 – L1 – I1

Table 5.7: Path selection for [TFRR + RR] model

We run a simulation with this model for a half an hour with the following setup.

- Clients send requests for web content application according to the profile configuration setup.
- The request will be served by one of the three content servers in round robin fashion.
- The server responses are routed along the LSPs assigned based on the paths chosen by TFRR algorithm, as shown in Table 5.7.
- The TCP load on each server and the traffic sent on each link were monitored.

Proposed CDN: TFRR and Server Selection based on residual capacities [TFRR + R1]

In this model, servers are selected for the requests originated in each subnets based on the updated residual capacities of the server following the server selection part of the TFRR algorithm. A server with maximum residual capacity is chosen to serve all the requests originated in a subnet. The server load is measured as the number of bps traffic generated by the server instead of the number of connections opened simultaneously, which is a simple load measure often used in CDN. Path is chosen based on the path selection part of the TFRR algorithm similar to [TFRR+RR] model.

The same network as shown in Figure 5.8 is used for simulating this model. In this model we perform request routing on Claps that is same server and paths are selected for all the subnets within a client network, as shown in Figure 5.6. For example, the subnet0 consists of five client subnets namely, Saskatoon, Toronto, Victoria, Winnipeg, and Yellowknife. Subnet1 also consists of five client subnets namely Atlanta, Boston, Chicago, Denver and Miami.

The expected load for each client network is given on Table 5.8, obtained from the simulation of conventional CDN. All servers have equal capacity and the maximum serving capacity is assumed to be 400kbps.

Network	Client Subnet	Load (bps)
Subnet0	Saskatoon	161,600
	Toronto	55,000
	Victoria	78,800
	Winnipeg	73,200
	Yellowknife	49,200
Subnet1	Atlanta	111,200
	Boston	62,400
	Chicago	110,000
	Denver	94,200
	Miami	16,200

Table 5.8: Expected load of client network

From the observations in the conventional CDN model, clients arrive or make content requests from the client networks in the following sequence: Winnipeg, Saskatoon, Atlanta, Chicago, Victoria, Yellowknife, Boston, Miami, Denver and Toronto.

The first content request comes from Winnipeg network, which is assigned a server and path based on the initial server and path selection steps of the TFRR algorithm. Initially, all servers have zero load so that the server with the maximum number of paths is chosen. Server2 has 7 available paths, Server1 has 4 paths and Server3 has 3 paths. Thus, Server2 is chosen. A path from Server2 to Subnet0 (Winnipeg) is chosen as discussed in TFRR+RR model. According to TFRR system, Server2 is assigned to serve the requests of all the clients in the Winnipeg network. The load from this client network is 73.2 kbps. The server loads are updated after making this selection to add 73.2 Kbps load to Server2 as shown in Table 5.9. :

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	0	161600	161600	238400	0.596
Server2	400000	73200	161600	234800	165200	0.413
Server3	400000	0	161600	161600	238400	0.596

Table 5.9: Expected load on servers

Similarly the average weight table and path weight tables are updated to reflect the traffic on the path selected for Winnipeg traffic, as shown in Table 5.10 and 5.11 respectively.

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.073	0.620
E3 – L3	0.214	1.000	0.000	1.214
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.073	1.144
L3 – I2	0.428	1.000	0.000	1.428

Table 5.10: Average weight table – [TFRR+RI] model

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.144		2.858
E1 – L2 – L3 – L1 – I1	0.786	0.928	1.001	1.144	3.859
E1 – L2 – L1 – L3 – I2	0.786	0.928	1.001	1.428	4.143
E1 – L2 – L3 – I2	0.786	0.928	1.428		3.142
E2 – L2 – L1 – I1	0.619	0.928	1.144		2.691
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.144	3.692
E2 – L3 – L1 – I1	0.620	1.001	1.144		2.766
E2 – L3 – L2 – L1 – I1	0.620	0.928	0.928	1.144	3.620
E2 – L3 – I2	0.620	1.428			2.048
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.001	1.428	3.976
E2 – L2 – L3 – I2	0.619	0.928	1.428		2.975
E3 – L3 – L2 – L1 – I1	1.214	0.928	0.928	1.144	4.214
E3 – L3 – L1 – I1	1.214	1.001	1.144		3.359
E3 – L3 – I2	1.214	1.428			2.642

Table 5.11: Path weight table – TFRR+RI model

Next, a request from Saskatoon network arrives to the CDN. The expected load of this network is 161.6k, as shown in Table 5.8. All servers have enough residual capacity to serve this request but both Server3 and Server1 have more residual capacities than Server2, because Server2 is assigned to serve Winnipeg clients in the previous step. Hence, either Server3 or Server1 can be chosen. In this case selection of the best path available in the network will identify which of the two servers should be selected. The process of path selection is identical with path selection in [TFRR + RI]. We first consider all the paths between Server3 and Saskatoon and Server1 and Saskatoon. The path E3 – L3 – I2 is the shortest path among all the paths considered, hence Server3 is chosen and the path is selected. All the clients in the Saskatoon network will be directed to this server. FEC is defined for the LSP along E3-L3-I2 for the CNAP corresponding to the Saskatoon network.

For clients in Atlanta network the above sequence of steps is repeated to assign a server and path. This process is repeated for selecting servers and paths for each of the client networks. The table 5.12 shows the server selection and path selection for each client network.

Subnet	Server	LSP	Path
Winnipeg	Server2	E2 – I1	E2 – L3 – L1 – I1
Saskatoon	Server3	E3 – I1	E3 – L3 – I2
Atlanta	Server1	E1 – I2	E1 – L2 – L1 – I1
Chicago	Server2	E2 – I2	E2 – L3 – I2
Victoria	Server1	E1 – I1	E1 – L2 – L1 – I1
Yellowknife	Server3	E3 – I2	E3 – L3 – I2
Boston	Server2	E2 – I2	E2 – L3 – I2
Miami	Server1	E1 – I1	E1 – L2 – L1 – I1
Denver	Server1	E1 – I1	E1 – L2 – L1 – I1
Toronto	Server3	E3 – I2	E3 – L3 – I2

Table 5.12: Server selection and path selection

We run the simulation for this model for half an hour with the following set up.

- Clients send requests for web content application according to the profile configuration setup.
- Requests of same client network will be served by one of the three content servers selected based on residual capacities under TFRR algorithm.
- The server responses are routed along the LSPs assigned based on the paths chosen by TFRR algorithm.
- The TCP load on each server and the traffic sent on each link were monitored.

5.4.3 RESULTS AND COMPARISONS

We present the results of simulating the above three models by considering content traffic originated at servers on each link and server load.

1. Server response traffic on each individual links.

There are altogether 9 links in the backbone network. In our discussion of the results we call links L1 – I1, L3 – I2 as ingress links, which connects the client networks to the backbone network. Links E1 – L2, E2 – L2, E2 – L3, E3 – L3 are called egress links, which connects the content servers to the backbone network. Links L2 – L1, L3 – L1, and L3 – L2 are called intermediate links.

2. Server load in terms bytes/sec.

Link Statistics on ingress links

Figure 5.10: Link Load Statistics on ingress links in bps.

Link Statistics on egress Links

Figure 5.11: Link Load Statistics on egress Links in bps.

Link Statistics on intermediate links

Figure 5.12: Link Load Statistics on intermediate links in bps.

Server Load Statistics

Figure 5.13: Server Load Statistics in bytes/sec.

Most of the traffic is routed through link L3 – I2 in the convention CDN model (IP Routing + RR) since it forms shortest paths for Server2 and Server3 and one of equal cost paths for Server1. This link is heavily congested, while link L1 – I1 remains under utilized. This is the typical problem in the conventional CDN network. Using high utilization links result in more packets loss, retransmissions, and long delays in packet transmission, which increases the access latency. In contrast, the TFRR algorithm sends traffic along the best paths to distribute the traffic load. The graphs corresponding to [TFRR+RR] model shows better link load distribution on the ingress links. The Proposed CDN [TFRR+RI] model graph still shows good load distribution but not better than TFRR+RR model. This is because in [TFRR+RI] model all clients of one particular network are routed along one path to improve processing efficiency and to achieve server load balancing, which causes skewed traffic distribution along the LSPs.

Traffic load on egress links follow the server load pattern, as they are directly connected to the servers. The first two models employs round robin sever selection, which shows poor load balancing among the servers. This also reflects as the poor traffic load distribution on egress links. In Proposed CDN [TFRR+RI] model server selection is made based on the TFRR algorithm, which improves load balancing among servers as shown in Figure 5.12. This also causes better traffic load distribution on the egress links, as shown in Figure 5.10. Notice that utilization of all the egress links are not uniform,

rather one of the egress link is never used in any model, which may be used when more number of client subnets send requests.

In the conventional CDN model, the intermediate links are also not used uniformly. Link L1 – L3 is never used; link L1 – L2 shows only peak traffic of nearly 175kbps, and average traffic over the period of time is approximately 100kbps. These links often belong to longer paths, which are seldom selected to carry the traffic. In TFRR+RR model, Links L1 – L2 and L1 – L3 show an average traffic of 250kbps each. In this model these links carry more traffic than in conventional CDN model but achieve good load balancing. Link L2 – L3 is rarely used in all three models. So the performance of that link is not compared. The third model also shows a considerable traffic on both links L1 – L2 and L1 – L3. But the load distribution is not better than in TFRR+RR model. This is offset by improving load distribution on egress links and server load balancing. In general, the uniform load distribution on egress links is more critical than the load distribution on intermediate links, because the intermediate links being core links have more available bandwidth.

The server load statistics shows that, server selection based on the TFRR algorithm performs better than server selection based on round robin quite commonly used by DNS. In summary, the TFRR system tries to achieve network load distribution and server load balancing at the same time.

The simulation results can be improved by using a wide backbone network with more number of available paths so that, there will be more flexibility in path selection. Hence the link load distribution will be further strengthened. Sending content request from as many as client subnets also will improve the results. When there is more number of client networks connects, there will be a number of path selections and server selections. So the load distribution will be further improved.

The performance of TFRR algorithm will be improved, when more number of client subnets send requests to content servers over a network with more number of paths

available. In reality, applying the TFRR algorithm can provide the advantage of server load balancing, network traffic load distribution together. Achieving these load distribution is vital to scale the system performance with increasing number of content requests demands.

We created another scenario, where the bandwidth of the link I2 – L3 is reduced to 256kbps while keeping the bandwidth of all other links to 1 Mbps, which make L2 — L3, a bottleneck link. The model used in the simulation is shown in Figure 5.14. Conventional CDN model and Proposed CDN [TFRR+RI] models are compared.

Figure 5.14: Simulation model

In conventional model, OSPF routing takes the shortest path regardless of the bandwidth constraint. Therefore, Server responses from Server2 and Server3 are routed through the shortest path that consists of the bottleneck link I2 – L3. The link statistics on ingress links for the conventional CDN clearly shows that the traffic reaches the capacity of I2—L3 link, 256kbps. The other ingress link I1—L1 that has a capacity of 1Mbps remains underutilized. The link statistics of TFRR+RI shows a better load distribution between the two ingress links as shown in Figure 5.15

IP Routing + RR

TFRR + RI

Figure 5.15: Link Load Statistics on ingress links in bps.

In conventional CDN, link I2 – L3 is highly congested and link I1 – L1 is underutilized. The queuing delay graph reflects this effect in Figure 5.16. For conventional CDN there is a very high queuing delay in link I2 – L3 and a very low queuing delay in link I1 – L1. Since there is better load distribution between the ingress links in case of TFRR+RI, the queuing delay of link I2 – L3 is much reduced. Also, Link I1 – L1 shows some queuing delay, indicating the link is utilized well.

Link I1 – L1

Link I2 – L3

Figure 5.16: Queuing delay of ingress links in seconds.

5.5 CHAPTER SUMMARY

This chapter explains the characteristics of a workload model and introduced a realistic workload model for content distribution. This kind of workload model is applied in the OPNET simulation. A feasible simulation setup using NS-2 software is explained with the results obtained. This simulation shows that load balancing scheme based on bandwidth proportions provides better server load balancing than a round robin scheme. At the same time, link utilization on the network is controlled well in bandwidth proportion scheme than in round robin scheme.

The TFRR system is tested using the OPNET simulation tool. Network performance of traditional network model, traffic engineering model using TFRR algorithm and content routing model using TFRR algorithm are examined. Results shows better traffic load distribution and server load balancing by using TFRR algorithm. By applying TFRR algorithm, server load balancing and network traffic load distribution can be achieved simultaneously.

6

Conclusion & Future Work

Achieving Server load balancing and network load distribution simultaneously in CDNs are very challenging task. A Content delivery architecture is discussed to achieve both server load balancing and network path load balancing. Content servers are grouped using a common IP anycast domain, while the path is chosen over an MPLS network based on constraint based routing and Traffic Engineering. A TE friendly request routing algorithm is proposed in this thesis to choose a content server for a client subnet with a higher residual capacity index and a path with least demand interference, least loaded and least criticality. Server with higher residual capacity is chosen to accommodate more number of future requests from the same subnet. A path is chosen such that it provides less interference to all other available paths for future requests and achieves network path load distribution.

Network performance is compared for round robin request routing scheme and proportional bandwidth request routing scheme. The proportional bandwidth scheme gives better load distribution and less latency in serving clients. The TFRR algorithm is simulated and its performance is compared with conventional CDN model. The simulation results show that, using the TFRR algorithm to perform request routing provides better server load balancing and network load distribution. Performance will be improved by simulating the algorithm with a large number of subnets and available paths. Better network performance can be achieved, as there are more flexibility in choosing the path and the server.

The algorithm makes server selection and path selection for active CNAPs and for newly arrived CNAPs, which access the content significantly. The algorithm can be extended to make a default path selection and server selection for CNAPs that arrive to the CDN for the first time. Further research work is required on how the TFRR algorithm is

implemented in the Decision Server of the proposed CDN architecture discussed in chapter3. This could show the realistic performance of a CDN. Further, analyzing the performance between applying TFRR algorithm in a Central Decision Server and applying it in a distributed manner should be explored. Processing times and overheads are significant limiting factors of this kind of algorithm. The impact of processing times and overheads on the network performance needs to be evaluated.

For our simulation, we computed the average weight compose of three weights using the constants a , b , c with the default value of 1. The impact of these constants can be analyzed by performing the simulation based on average weights computed using different values for the constants.

APPENDIX 1

Traffic Load per Client (Bps)

Client/ Server	Server1	Server2	Server3
S1	775		
S2			
S3			1600
S4	5400		
S5		12425	
T1			2100
T2			
T3		3875	
T4			900
T5			
V1		1200	
V2			
V3			
V4		8150	
V5			500
W1	3300		
W2		1200	
W3			3750
W4			
W5		900	
Y1			750
Y2	1700		
Y3		3700	
Y4			
Y5			
Total (Bps)	11175	31450	9600
Total (bps)	89400	251600	76800

Client/ Server	Server1	Server2	Server3
A1			
A2			3550
A3	3375		
A4			
A5			6975
B1	1600		
B2		700	
B3			
B4	5500		
B5			
C1			1300
C2			
C3		3500	
C4			8000
C5	950		
D1		1325	
D2			3150
D3			
D4		5600	
D5			1700
M1	1250		
M2			
M3			775
M4			
M5			
Total (BPS)	12675	11125	25450
Total (bps)	101400	89000	203600

APPENDIX 2

Path Selection for Proposed CDN [TFRR + RR]

1. Initial Path Selection for (Server2 – Subnet0)

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.000	0.547
E3 – L3	0.214	1.000	0.000	1.214
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.000	0.928
L1 – I1	0.571	0.500	0.000	1.071
L3 – I2	0.428	1.000	0.000	1.428

Path
E2 – L3 – L1 – I1

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.071		2.785
E1 – L2 – L3 – L1 – I1	0.786	0.928	0.928	1.071	3.713
E1 – L2 – L1 – L3 – I2	0.786	0.928	0.928	1.428	4.070
E1 – L2 – L3 – I2	0.786	0.928	1.428		3.142
E2 – L2 – L1 – I1	0.619	0.928	1.071		2.618
E2 – L2 – L3 – L1 – I1	0.619	0.928	0.928	1.071	3.546
E2 – L3 – L1 – I1	0.547	0.928	1.071		2.546
E2 – L3 – L2 – L1 – I1	0.547	0.928	0.928	1.071	3.474
E2 – L3 – I2	0.547	1.428			1.975
E2 – L2 – L1 – L3 – I2	0.619	0.928	0.928	1.428	3.903
E2 – L2 – L3 – I2	0.619	0.928	1.428		2.975
E3 – L3 – L2 – L1 – I1	1.214	0.928	0.928	1.071	4.141
E3 – L3 – L1 – I1	1.214	0.928	1.071		3.213
E3 – L3 – I2	1.214	1.428			2.642

2. Path Selection for (Server3 – Subnet1)

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.252	0.799
E3 – L3	0.214	1.000	0.000	1.214
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.252	1.180
L1 – I1	0.571	0.500	0.252	1.323
L3 – I2	0.428	1.000	0.000	1.428

Path
E3 – L3 – I2

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.323		3.037
E1 – L2 – L3 – L1 – I1	0.786	0.928	1.180	1.323	4.216
E1 – L2 – L1 – L3 – I2	0.786	0.928	1.180	1.428	4.322
E1 – L2 – L3 – I2	0.786	0.928	1.428		3.142
E2 – L2 – L1 – I1	0.619	0.928	1.323		2.870
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.180	1.323	4.049
E2 – L3 – L1 – I1	0.799	1.180	1.323		3.301
E2 – L3 – L2 – L1 – I1	0.799	0.928	0.928	1.323	3.977
E2 – L3 – I2	0.799	1.428			2.227
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.180	1.428	4.155
E2 – L2 – L3 – I2	0.619	0.928	1.428		2.975
E3 – L3 – L2 – L1 – I1	1.214	0.928	0.928	1.323	4.393
E3 – L3 – L1 – I1	1.214	1.180	1.323		3.716
E3 – L3 – I2	1.214	1.428			2.642

3. Path Selection for (Server1 – Subnet1)

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.252	0.799
E3 – L3	0.214	1.000	0.204	1.418
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.252	1.180
L1 – I1	0.571	0.500	0.252	1.323
L3 – I2	0.428	1.000	0.204	1.632

Path
E1 – L2 – L3 – I2

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.323		3.037
E1 – L2 – L3 – L1 – I1	0.786	0.928	1.180	1.323	4.216
E1 – L2 – L1 – L3 – I2	0.786	0.928	1.180	1.632	4.525
E1 – L2 – L3 – I2	0.786	0.928	1.632		3.346
E2 – L2 – L1 – I1	0.619	0.928	1.323		2.870
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.180	1.323	4.049
E2 – L3 – L1 – I1	0.799	1.180	1.323		3.301
E2 – L3 – L2 – L1 – I1	0.799	0.928	0.928	1.323	3.977
E2 – L3 – I2	0.799	1.632			2.430
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.180	1.632	4.358
E2 – L2 – L3 – I2	0.619	0.928	1.632		3.179
E3 – L3 – L2 – L1 – I1	1.418	0.928	0.928	1.323	4.596
E3 – L3 – L1 – I1	1.418	1.180	1.323		3.920
E3 – L3 – I2	1.418	1.632			3.049

4. Path Selection for (Server1 – Subnet0)

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.101	0.887
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.252	0.799
E3 – L3	0.214	1.000	0.204	1.418
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.101	1.029
L3 – L1	0.428	0.500	0.252	1.180
L1 – I1	0.571	0.500	0.252	1.323
L3 – I2	0.428	1.000	0.305	1.733

Path
E1 – L2 – L1 – I1

Path					Weight
E1 – L2 – L1 – I1	0.887	0.928	1.323		3.138
E1 – L2 – L3 – L1 – I1	0.887	1.029	1.180	1.323	4.419
E1 – L2 – L1 – L3 – I2	0.887	0.928	1.180	1.733	4.728
E1 – L2 – L3 – I2	0.887	1.029	1.733		3.650
E2 – L2 – L1 – I1	0.619	0.928	1.323		2.870
E2 – L2 – L3 – L1 – I1	0.619	1.029	1.180	1.323	4.151
E2 – L3 – L1 – I1	0.799	1.180	1.323		3.301
E2 – L3 – L2 – L1 – I1	0.799	1.029	0.928	1.323	4.079
E2 – L3 – I2	0.799	1.733			2.532
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.180	1.733	4.460
E2 – L2 – L3 – I2	0.619	1.029	1.733		3.381
E3 – L3 – L2 – L1 – I1	1.418	1.029	0.928	1.323	4.698
E3 – L3 – L1 – I1	1.418	1.180	1.323		3.920
E3 – L3 – I2	1.418	1.733			3.151

5. Path Selection for (Server2– Subnet1)

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.191	0.977
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.252	0.799
E3 – L3	0.214	1.000	0.204	1.418
L2 – L1	0.428	0.500	0.089	1.017
L3 – L2	0.428	0.500	0.101	1.029
L3 – L1	0.428	0.500	0.252	1.180
L1 – I1	0.571	0.500	0.341	1.412
L3 – I2	0.428	1.000	0.305	1.733

Path
E2 – L3 – I2

Path					Weight
E1 – L2 – L1 – I1	0.977	1.017	1.412		3.406
E1 – L2 – L3 – L1 – I1	0.977	1.029	1.180	1.412	4.598
E1 – L2 – L1 – L3 – I2	0.977	1.017	1.180	1.733	4.907
E1 – L2 – L3 – I2	0.977	1.029	1.733		3.739
E2 – L2 – L1 – I1	0.619	1.017	1.412		3.048
E2 – L2 – L3 – L1 – I1	0.619	1.029	1.180	1.412	4.240
E2 – L3 – L1 – I1	0.799	1.180	1.412		3.390
E2 – L3 – L2 – L1 – I1	0.799	1.029	1.017	1.412	4.257
E2 – L3 – I2	0.799	1.733			2.532
E2 – L2 – L1 – L3 – I2	0.619	1.017	1.180	1.733	4.549
E2 – L2 – L3 – I2	0.619	1.029	1.733		3.381
E3 – L3 – L2 – L1 – I1	1.418	1.029	1.017	1.412	4.876
E3 – L3 – L1 – I1	1.418	1.180	1.412		4.009
E3 – L3 – I2	1.418	1.733			3.151

6. Path Selection for (Server3 – Subnet0)

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.191	0.977
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.252	0.799
E3 – L3	0.214	1.000	0.293	1.507
L2 – L1	0.428	0.500	0.089	1.017
L3 – L2	0.428	0.500	0.101	1.029
L3 – L1	0.428	0.500	0.252	1.180
L1 – I1	0.571	0.500	0.341	1.412
L3 – I2	0.428	1.000	0.394	1.822

Path
E3 – L3 – L1 – I1

Path					Weight
E1 – L2 – L1 – I1	0.977	1.017	1.412		3.406
E1 – L2 – L3 – L1 – I1	0.977	1.029	1.180	1.412	4.598
E1 – L2 – L1 – L3 – I2	0.977	1.017	1.180	1.822	4.996
E1 – L2 – L3 – I2	0.977	1.029	1.822		3.828
E2 – L2 – L1 – I1	0.619	1.017	1.412		3.048
E2 – L2 – L3 – L1 – I1	0.619	1.029	1.180	1.412	4.240
E2 – L3 – L1 – I1	0.799	1.180	1.412		3.390
E2 – L3 – L2 – L1 – I1	0.799	1.029	1.017	1.412	4.257
E2 – L3 – I2	0.799	1.822			2.621
E2 – L2 – L1 – L3 – I2	0.619	1.017	1.180	1.822	4.638
E2 – L2 – L3 – I2	0.619	1.029	1.822		3.470
E3 – L3 – L2 – L1 – I1	1.507	1.029	1.017	1.412	4.965
E3 – L3 – L1 – I1	1.507	1.180	1.412		4.098
E3 – L3 – I2	1.507	1.822			3.329

APPENDIX 3

Server Selection and Path Selection for Proposed CDN [TFRR + RI]

1. Initial Server and Path Selection for Winnipeg Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.000	0.547
E3 – L3	0.214	1.000	0.000	1.214
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.000	0.928
L1 – I1	0.571	0.500	0.000	1.071
L3 – I2	0.428	1.000	0.000	1.428

Path / Server
E2 – L3 – L1 – I1 / Server2

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.071		2.785
E1 – L2 – L3 – L1 – I1	0.786	0.928	0.928	1.071	3.713
E1 – L2 – L1 – L3 – I2	0.786	0.928	0.928	1.428	4.070
E1 – L2 – L3 – I2	0.786	0.928	1.428		3.142
E2 – L2 – L1 – I1	0.619	0.928	1.071		2.618
E2 – L2 – L3 – L1 – I1	0.619	0.928	0.928	1.071	3.546
E2 – L3 – L1 – I1	0.547	0.928	1.071		2.546
E2 – L3 – L2 – L1 – I1	0.547	0.928	0.928	1.071	3.474
E2 – L3 – I2	0.547	1.428			1.975
E2 – L2 – L1 – L3 – I2	0.619	0.928	0.928	1.428	3.903
E2 – L2 – L3 – I2	0.619	0.928	1.428		2.975
E3 – L3 – L2 – L1 – I1	1.214	0.928	0.928	1.071	4.141
E3 – L3 – L1 – I1	1.214	0.928	1.071		3.213
E3 – L3 – I2	1.214	1.428			2.642

2. Server and Path Selection for Saskatoon Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.073	0.620
E3 – L3	0.214	1.000	0.000	1.214
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.073	1.144
L3 – I2	0.428	1.000	0.000	1.428

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.144		2.858
E1 – L2 – L3 – L1 – I1	0.786	0.928	1.001	1.144	3.859
E1 – L2 – L1 – L3 – I2	0.786	0.928	1.001	1.428	4.143
E1 – L2 – L3 – I2	0.786	0.928	1.428		3.142
E2 – L2 – L1 – I1	0.619	0.928	1.144		2.691
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.144	3.692
E2 – L3 – L1 – I1	0.620	1.001	1.144		2.766
E2 – L3 – L2 – L1 – I1	0.620	0.928	0.928	1.144	3.620
E2 – L3 – I2	0.620	1.428			2.048
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.001	1.428	3.976
E2 – L2 – L3 – I2	0.619	0.928	1.428		2.975
E3 – L3 – L2 – L1 – I1	1.214	0.928	0.928	1.144	4.214
E3 – L3 – L1 – I1	1.214	1.001	1.144		3.359
E3 – L3 – I2	1.214	1.428			2.642

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	0	161600	161600	238400	0.596
Server2	400000	73200	161600	234800	165200	0.413
Server3	400000	0	161600	161600	238400	0.596

Path / Server
E3 – L3 – I2 / Server3

3. Server and Path Selection for Atlanta Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.000	0.786
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.073	0.620
E3 – L3	0.214	1.000	0.162	1.376
L2 – L1	0.428	0.500	0.000	0.928
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.073	1.144
L3 – I2	0.428	1.000	0.162	1.590

Path					Weight
E1 – L2 – L1 – I1	0.786	0.928	1.144		2.858
E1 – L2 – L3 – L1 – I1	0.786	0.928	1.001	1.144	3.859
E1 – L2 – L1 – L3 – I2	0.786	0.928	1.001	1.590	4.305
E1 – L2 – L3 – I2	0.786	0.928	1.590		3.304
E2 – L2 – L1 – I1	0.619	0.928	1.144		2.691
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.144	3.692
E2 – L3 – L1 – I1	0.620	1.001	1.144		2.766
E2 – L3 – L2 – L1 – I1	0.620	0.928	0.928	1.144	3.620
E2 – L3 – I2	0.620	1.590			2.210
E2 – L2 – L1 – L3 – I2	0.619	0.928	1.001	1.590	4.138
E2 – L2 – L3 – I2	0.619	0.928	1.590		3.137
E3 – L3 – L2 – L1 – I1	1.376	0.928	0.928	1.144	4.376
E3 – L3 – L1 – I1	1.376	1.001	1.144		3.521
E3 – L3 – I2	1.376	1.590			2.965

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	0	111200	111200	288800	0.722
Server2	400000	73200	111200	184400	215600	0.539
Server3	400000	161600	111200	272800	127200	0.318

Path / Server
E1 – L2 – L1 – I1
/ Server1

4. Server and Path Selection for Chicago Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.111	0.897
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.073	0.620
E3 – L3	0.214	1.000	0.162	1.376
L2 – L1	0.428	0.500	0.111	1.039
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.184	1.255
L3 – I2	0.428	1.000	0.162	1.590

Path					Weight
E1 – L2 – L1 – I1	0.897	1.039	1.255		3.192
E1 – L2 – L3 – L1 – I1	0.897	0.928	1.001	1.255	4.082
E1 – L2 – L1 – L3 – I2	0.897	1.039	1.001	1.590	4.527
E1 – L2 – L3 – I2	0.897	0.928	1.590		3.415
E2 – L2 – L1 – I1	0.619	1.039	1.255		2.914
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.255	3.804
E2 – L3 – L1 – I1	0.620	1.001	1.255		2.877
E2 – L3 – L2 – L1 – I1	0.620	0.928	1.039	1.255	3.843
E2 – L3 – I2	0.620	1.590			2.210
E2 – L2 – L1 – L3 – I2	0.619	1.039	1.001	1.590	4.249
E2 – L2 – L3 – I2	0.619	0.928	1.590		3.137
E3 – L3 – L2 – L1 – I1	1.376	0.928	1.039	1.255	4.598
E3 – L3 – L1 – I1	1.376	1.001	1.255		3.632
E3 – L3 – I2	1.376	1.590			2.965

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	111200	110000	221200	178800	0.447
Server2	400000	73200	110000	183200	216800	0.542
Server3	400000	161600	110000	271600	128400	0.321

Path / Server
E2 – L3 – I2
/ Server2

5. Server and Path Selection for Victoria Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.111	0.897
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.183	0.730
E3 – L3	0.214	1.000	0.162	1.376
L2 – L1	0.428	0.500	0.111	1.039
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.184	1.255
L3 – I2	0.428	1.000	0.272	1.700

Path					Weight
E1 – L2 – L1 – I1	0.897	1.039	1.255		3.192
E1 – L2 – L3 – L1 – I1	0.897	0.928	1.001	1.255	4.082
E1 – L2 – L1 – L3 – I2	0.897	1.039	1.001	1.700	4.637
E1 – L2 – L3 – I2	0.897	0.928	1.700		3.525
E2 – L2 – L1 – I1	0.619	1.039	1.255		2.914
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.255	3.804
E2 – L3 – L1 – I1	0.730	1.001	1.255		2.987
E2 – L3 – L2 – L1 – I1	0.730	0.928	1.039	1.255	3.953
E2 – L3 – I2	0.730	1.700			2.430
E2 – L2 – L1 – L3 – I2	0.619	1.039	1.001	1.700	4.359
E2 – L2 – L3 – I2	0.619	0.928	1.700		3.247
E3 – L3 – L2 – L1 – I1	1.376	0.928	1.039	1.255	4.598
E3 – L3 – L1 – I1	1.376	1.001	1.255		3.632
E3 – L3 – I2	1.376	1.700			3.075

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	111200	78800	190000	210000	0.525
Server2	400000	183200	78800	262000	138000	0.345
Server3	400000	161600	78800	240400	159600	0.399

Path / Server
E1 – L2 – L1 – I1
/ Server1

6. Server and Path Selection for Yellowknife Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.190	0.976
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.183	0.730
E3 – L3	0.214	1.000	0.162	1.376
L2 – L1	0.428	0.500	0.190	1.118
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.263	1.334
L3 – I2	0.428	1.000	0.272	1.700

Path					Weight
E1 – L2 – L1 – I1	0.976	1.118	1.334		3.428
E1 – L2 – L3 – L1 – I1	0.976	0.928	1.001	1.334	4.239
E1 – L2 – L1 – L3 – I2	0.976	1.118	1.001	1.700	4.795
E1 – L2 – L3 – I2	0.976	0.928	1.700		3.604
E2 – L2 – L1 – I1	0.619	1.118	1.334		3.071
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.334	3.882
E2 – L3 – L1 – I1	0.730	1.001	1.334		3.066
E2 – L3 – L2 – L1 – I1	0.730	0.928	1.118	1.334	4.110
E2 – L3 – I2	0.730	1.700			2.430
E2 – L2 – L1 – L3 – I2	0.619	1.118	1.001	1.700	4.438
E2 – L2 – L3 – I2	0.619	0.928	1.700		3.247
E3 – L3 – L2 – L1 – I1	1.376	0.928	1.118	1.334	4.756
E3 – L3 – L1 – I1	1.376	1.001	1.334		3.711
E3 – L3 – I2	1.376	1.700			3.075

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	190000	49200	239200	160800	0.402
Server2	400000	183200	49200	232400	167600	0.419
Server3	400000	161600	49200	210800	189200	0.473

Path / Server
E3 – L3 – I2
/ Server3

7. Server and Path Selection for Boston Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.190	0.976
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.183	0.730
E3 – L3	0.214	1.000	0.211	1.425
L2 – L1	0.428	0.500	0.190	1.118
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.263	1.334
L3 – I2	0.428	1.000	0.321	1.749

Path					Weight
E1 – L2 – L1 – I1	0.976	1.118	1.334		3.428
E1 – L2 – L3 – L1 – I1	0.976	0.928	1.001	1.334	4.239
E1 – L2 – L1 – L3 – I2	0.976	1.118	1.001	1.749	4.844
E1 – L2 – L3 – I2	0.976	0.928	1.749		3.653
E2 – L2 – L1 – I1	0.619	1.118	1.334		3.071
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.334	3.882
E2 – L3 – L1 – I1	0.730	1.001	1.334		3.066
E2 – L3 – L2 – L1 – I1	0.730	0.928	1.118	1.334	4.110
E2 – L3 – I2	0.730	1.749			2.479
E2 – L2 – L1 – L3 – I2	0.619	1.118	1.001	1.749	4.487
E2 – L2 – L3 – I2	0.619	0.928	1.749		3.296
E3 – L3 – L2 – L1 – I1	1.425	0.928	1.118	1.334	4.805
E3 – L3 – L1 – I1	1.425	1.001	1.334		3.760
E3 – L3 – I2	1.425	1.749			3.174

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	190000	62400	252400	147600	0.369
Server2	400000	183200	62400	245600	154400	0.386
Server3	400000	210800	62400	273200	126800	0.317

Path / Server
E2 – L3 – I2
/ Server2

8. Server and Path Selection for Miami Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.190	0.976
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.246	0.793
E3 – L3	0.214	1.000	0.211	1.425
L2 – L1	0.428	0.500	0.190	1.118
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.263	1.334
L3 – I2	0.428	1.000	0.383	1.811

Path					Weight
E1 – L2 – L1 – I1	0.976	1.118	1.334		3.428
E1 – L2 – L3 – L1 – I1	0.976	0.928	1.001	1.334	4.239
E1 – L2 – L1 – L3 – I2	0.976	1.118	1.001	1.811	4.906
E1 – L2 – L3 – I2	0.976	0.928	1.811		3.715
E2 – L2 – L1 – I1	0.619	1.118	1.334		3.071
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.334	3.882
E2 – L3 – L1 – I1	0.793	1.001	1.334		3.128
E2 – L3 – L2 – L1 – I1	0.793	0.928	1.118	1.334	4.173
E2 – L3 – I2	0.793	1.811			2.604
E2 – L2 – L1 – L3 – I2	0.619	1.118	1.001	1.811	4.549
E2 – L2 – L3 – I2	0.619	0.928	1.811		3.358
E3 – L3 – L2 – L1 – I1	1.425	0.928	1.118	1.334	4.805
E3 – L3 – L1 – I1	1.425	1.001	1.334		3.760
E3 – L3 – I2	1.425	1.811			3.236

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	190000	16200	206200	193800	0.485
Server2	400000	245600	16200	261800	138200	0.346
Server3	400000	210800	16200	227000	173000	0.433

Path / Server
E1 – L2 – L1 – I1
/ Server1

9. Server and Path Selection for Denver Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.206	0.992
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.246	0.793
E3 – L3	0.214	1.000	0.211	1.425
L2 – L1	0.428	0.500	0.206	1.134
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.279	1.350
L3 – I2	0.428	1.000	0.383	1.811

Path					Weight
E1 – L2 – L1 – I1	0.992	1.134	1.350		3.477
E1 – L2 – L3 – L1 – I1	0.992	0.928	1.001	1.350	4.272
E1 – L2 – L1 – L3 – I2	0.992	1.134	1.001	1.811	4.939
E1 – L2 – L3 – I2	0.992	0.928	1.811		3.731
E2 – L2 – L1 – I1	0.619	1.134	1.350		3.104
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.350	3.899
E2 – L3 – L1 – I1	0.793	1.001	1.350		3.144
E2 – L3 – L2 – L1 – I1	0.793	0.928	1.134	1.350	4.205
E2 – L3 – I2	0.793	1.811			2.604
E2 – L2 – L1 – L3 – I2	0.619	1.134	1.001	1.811	4.566
E2 – L2 – L3 – I2	0.619	0.928	1.811		3.358
E3 – L3 – L2 – L1 – I1	1.425	0.928	1.134	1.350	4.837
E3 – L3 – L1 – I1	1.425	1.001	1.350		3.776
E3 – L3 – I3	1.425	1.811			3.236

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	206200	94200	300400	99600	0.249
Server2	400000	245600	94200	339800	60200	0.151
Server3	400000	210800	94200	305000	95000	0.238

Path / Server
E1 – L2 – L1 – I1
/ Server1

10. Server and Path Selection for Toronto Network

Link	DW	CW	UW	AW
E1 – L2	0.286	0.500	0.300	1.086
E2 – L2	0.286	0.333	0.000	0.619
E2 – L3	0.214	0.333	0.246	0.793
E3 – L3	0.214	1.000	0.211	1.425
L2 – L1	0.428	0.500	0.300	1.228
L3 – L2	0.428	0.500	0.000	0.928
L3 – L1	0.428	0.500	0.073	1.001
L1 – I1	0.571	0.500	0.374	1.445
L3 – I2	0.428	1.000	0.383	1.811

Path					Weight
E1 – L2 – L1 – I1	1.086	1.228	1.445		3.759
E1 – L2 – L3 – L1 – I1	1.086	0.928	1.001	1.445	4.460
E1 – L2 – L1 – L3 – I2	1.086	1.228	1.001	1.811	5.127
E1 – L2 – L3 – I2	1.086	0.928	1.811		3.826
E2 – L2 – L1 – I1	0.619	1.228	1.445		3.292
E2 – L2 – L3 – L1 – I1	0.619	0.928	1.001	1.445	3.993
E2 – L3 – L1 – I1	0.793	1.001	1.445		3.238
E2 – L3 – L2 – L1 – I1	0.793	0.928	1.228	1.445	4.394
E2 – L3 – I2	0.793	1.811			2.604
E2 – L2 – L1 – L3 – I2	0.619	1.228	1.001	1.811	4.660
E2 – L2 – L3 – I2	0.619	0.928	1.811		3.358
E3 – L3 – L2 – L1 – I1	1.425	0.928	1.228	1.445	5.026
E3 – L3 – L1 – I1	1.425	1.001	1.445		3.871
E3 – L3 – I3	1.425	1.811			3.236

Expected Load

Server	Capacity	Load	Exp. Load	Total Load	Residual	RI
Server1	400000	300400	55000	355400	44600	0.112
Server2	400000	245600	55000	300600	99400	0.249
Server3	400000	210800	55000	265800	134200	0.336

Path / Server
E3 – L3 – I3
/ Server3

APPENDIX 4

Probability Distributions

1. Poisson Distribution

Parameters $\lambda \geq 0$

Support $k \in \{0, 1, 2, \dots\}$

pmf $\frac{e^{-\lambda} \lambda^k}{k!}$

cdf $\frac{\Gamma(k+1, \lambda)}{k!}$

Mean λ

Median N/A

Mode $\lfloor \lambda \rfloor$

Variance λ

Skewness $\lambda^{-1/2}$

Kurtosis λ^{-1}

Entropy $\lambda[1 - \ln(\lambda)] + e^{-\lambda} \sum_{k=0}^{\infty} \frac{\lambda^k \ln(k!)}{k!}$

mgf $\exp(\lambda(e^t - 1))$

Char. func. $\exp(\lambda(e^{it} - 1))$

2. Zipf Distribution

Probability mass function

Cumulative distribution function

Parameters $s > 0$ (real)
 $N \in \{1, 2, 3, \dots\}$ (integer)

Support $k \in \{0, 1, 2, \dots, N\}$

$$\text{pdf} \quad \frac{1/k^s}{H_{N,s}}$$

$$\text{cdf} \quad \frac{H_{k,s}}{H_{N,s}}$$

$$\text{Mean} \quad \frac{H_{N,s-1}}{H_{N,s}}$$

Median N/A

$$\text{Mode} \quad \frac{1}{H_{N,s}}$$

$$\text{Entropy} \quad \frac{s}{H_{N,s}} \sum_{k=1}^N \frac{\ln(k)}{k^s} + \ln(H_{N,s})$$

$$\text{mf} \quad \frac{1}{H_{N,s}} \sum_{n=1}^N \frac{e^{nt}}{n^s}$$

$$\text{Char. func.} \quad \frac{1}{H_{N,s}} \sum_{n=1}^N \frac{e^{int}}{n^s}$$

3. Log-normal Distribution

Parameters $s \geq 0$
 $-\infty \leq \mu \leq \infty$

Support $x \in [0; +\infty)$

PDF
$$\frac{e^{-[\frac{\ln(x)-\mu}{\sigma}]^2/2}}{x\sigma\sqrt{2\pi}}$$

CDF
$$\frac{1}{2} + \frac{1}{2} \text{Erf} \left[\frac{\ln(x) - \mu}{\sigma\sqrt{2}} \right]$$

Mean $e^{\mu+\sigma^2/2}$

Median e^{μ}

Mode $e^{\mu-\sigma^2}$

Variance $(e^{\sigma^2} - 1)e^{2\mu+\sigma^2}$

Skewness $e^{-\mu-\sigma^2/2}(e^{\sigma^2}+2)\sqrt{e^{\sigma^2}-1}$

Kurtosis $e^{4\sigma^2} + 2e^{3\sigma^2} + 3e^{2\sigma^2} - 6$

Entropy $\frac{1}{2} + \frac{1}{2} \ln(2\pi)$

4. Pareto Distribution

Pareto probability density functions for various k with $x_m = 1$. The horizontal axis is the x parameter. Note that as $k \rightarrow \infty$ the distribution approaches $\delta(x - x_m)$ where δ is the Dirac delta function.

Pareto cumulative density functions for various k with $x_m = 1$. The horizontal axis is the x parameter.

Parameters $x_m > 0$ location (real)
 $k > 0$ shape (real)

Support $x \in [x_m; +\infty)$

pdf $\frac{k x_m^k}{x^{k+1}}$

cdf $1 - \left(\frac{x_m}{x}\right)^k$

Mean $\frac{k x_m}{k-1}$ for $k > 1$

Median $x_m \sqrt[k]{2}$

Mode x_m

Variance $\frac{x_m^2 k}{(k-1)^2(k-2)}$ for $k > 2$

Skewness $\frac{2(1+k)}{k-3} \sqrt{\frac{k-2}{k}}$ for $k > 3$

Kurtosis $\frac{6(k^3 + k^2 - 6k - 2)}{k(k-3)(k-4)}$ for $k > 4$

Entropy $\ln\left(\frac{k}{x_m}\right) - \frac{1}{k} - 1$

Char. func. $k(-ix_m t)^k \Gamma(-k, -ix_m t)$

5. Exponential Distribution

Parameters $\lambda > 0$ rate or inverse scale (real)

Support $x \in [0; \infty)$

pdf $\lambda e^{-\lambda x}$

cdf $1 - e^{-\lambda x}$

Mean λ^{-1}

Median $\ln(2)/\lambda$

Mode 0

Variance λ^{-2}

Skewness 2

Kurtosis 6

Entropy $1 - \ln(\lambda)$

mgf $\left(1 - \frac{t}{\lambda}\right)^{-1}$

Char. func. $\left(1 - \frac{it}{\lambda}\right)^{-1}$

Source: http://en.wikipedia.org/wiki/Main_Page

References

- [1] A. Acharya, A. Shaikh, R. Tewari, and D. Verma, "Scalable Web Request Routing with MPLS," IBM Research Report, December 2001.
- [2] R. Bonica, D. C. Tappan, "ICMP Extensions for Multi Protocol Label Switching", Internet Draft, February 1999.
- [3] A. Botta, P. Iovanna, R. Mameli, G. Piantanida, S. Salsano, "Extending MPLS Traffic Engineering to deal with QoS", *SoftCOM*, Split, Venezia, Ancona, Dubrovnik, October 2003.
- [4] X. Chang, "Network Simulations with OPNET", Proceedings of Winter Simulation Conference, 1999.
- [5] Cisco Systems, www.cisco.com
- [6] R. Dragos, S. Dragos, and M. Collier, "Design and implementation of an MPLS based load balancing architecture for Web switching", Proceedings of 15th ITC Specialist Seminar, Wurzburg, Germany, July 2002.
- [7] R. Engel, V. Peris, D. Saha, E. Bastruk, and R. Hass, "Using IP anycast for load distribution and server location", Proceedings of Global Internet, December 1998.
- [8] Z. Fu, N. Venkatasubramanian, "Combined Path and Server Selection in Dynamic Multimedia Environments", *ACM Multimedia*, November 1999.
- [9] A. Ghawani, B. Jamoussi, D. Fedyk, P. Ashwood-Smith, L. Li, N. Feldman, "Traffic Engineering Standards in IP Networks Using MPLS", *IEEE Communications Magazine*, Dec. 1999.
- [10] M. Jaseemuddin, A. Nanthakumaran, A. Leon-Garcia, Content Delivery Network Architecture Employing TE-Friendly Request Routing, Technical Report.
- [11] M. Kodialam, T. V. Lakshman, "Minimum Interference Routing with Applications to MPLS Traffic Engineering", *INFOCOMM 2000*.
- [12] M. Masa, E. Parravicini, "Impact of Request Routing Algorithms on the Delivery Performance of Content Delivery Networks." *IEEE Magazine*, 2003.

- [13] C. Metz, "IP Anycast: Point to (Any) Point Communications", Draft 0.3, Cisco Systems.
- [14] NS-2 Network Simulator: <http://www.isi.edu/nsnam/ns/>
- [15] OPNET, <http://www.opnet.com>
- [16] C. Partridge, T. Mendez, W. Milliken, "Host Anycasting Service", RFC 1546, November 1993.
- [17] W. Sa-Ngiamsak, R. Varakulsiripunth, "A Bandwidth-Based Constraint Routing Algorithm for Multi-Protocol Label Switching Networks", IEEE ICACT2004, 2004.
- [18] A. Shaikh, R. Tewari, M. Agrawal, "On the Effectiveness of DNS-based Server Selection", INFOCOMM 2001.
- [19] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet", IMC'04, October 2004.
- [20] E. Veloso, V. Almeida, W. Meira, A. Bestavros, S. Jin, "A Hierarchical Characterization of a Live Streaming Media Workload", ACM SIGCOMM, November 2002.
- [21] X. Xiao, A. Hannan, B. Bailey, L. M. Ni, "Traffic Engineering with MPLS in the Internet", IEEE Network Magazine, Mar. 2000.
- [22] M. Yamamoto, H. Miura, K. Nishimura, H. Ikeda, "A Network-Supported Server Load Balancing Method: Active Anycast", Paper-special issue on New Developments on QoS Technologies for Information Networks, November 2000.
- [23] O. Younis, S. Fahmy, "Constraint-Based Routing in the Internet: Basic Principles and Recent Research", IEEE Communications Surveys and Tutorials, 3rd quarter 2003.