Utility-Oriented Internetworking of Content Delivery Networks

by

Al-Mukaddim Khan Pathan

Submitted in total fulfilment of
the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Software Engineering
The University of Melbourne, Australia

November 2009

Produced on archival quality paper

Utility-Oriented Internetworking of Content Delivery Networks

Al-Mukaddim Khan Pathan

Supervisor: Professor Rajkumar Buyya

Abstract

Today's Internet content providers primarily use Content Delivery Networks (CDNs) to deliver content to end-users with the aim to enhance their Web access experience. Yet the prevalent commercial CDNs, operating in isolation, often face resource over-provisioning, degraded performance, and Service Level Agreement (SLA) violations, thus incurring high operational costs and limiting the scope and scale of their services.

To move beyond these shortcomings, this thesis sets out to establish the basis for developing advanced and efficient content delivery solutions that are scalable, high performance, and cost-effective. It introduces techniques to enable coordination and cooperation between multiple content delivery services, which is termed as "CDN peering". In this context, this thesis addresses five key issues—*when to peer* (triggering circumstances), *how to peer* (interaction strategies), *whom to peer with* (resource discovery), *how to manage and enforce operational policies* (request-redirection and load sharing), and *how to demonstrate peering applicability* (measurement study and proof-of-concept implementation).

Thesis Contributions: To support the thesis that the resource over-provisioning and degraded performance problems of existing CDNs can be overcome, thus improving Web access experience of Internet end-users, we have:

- identified the key research challenges and core technical issues for CDN peering, along with a systematic understanding of the CDN space by covering relevant applications, features and implementation techniques, captured in a comprehensive taxonomy of CDNs;
- developed a novel architectural framework, which provides the basis for CDN peering, formed by a set of autonomous CDNs that cooperate through an interconnection mechanism, providing the infrastructure and facilities to virtualize the service of multiple providers;
- devised Quality-of-Service (QoS)-oriented analytical performance models to demonstrate the effects of CDN peering and predict end-user perceived performance, thus facilitating to make concrete QoS performance guarantees for a CDN provider;
- developed enabling techniques, i.e. resource discovery, server selection, and request-redirection algorithms, for CDN peering to achieve service

responsiveness. These techniques are exercised to alleviate imbalanced load conditions, while minimizing redirection cost;

- introduced a utility model for CDN peering to measure its content-serving ability by capturing the traffic activities in the system and evaluated through extensive discrete-event simulation analysis. The findings of this study provide incentive for the exploitation of critical parameters for a better CDN peering system design; and
- demonstrated a *proof-of-concept* implementation of the utility model and an empirical measurement study on MetaCDN, which is a global overlay for Cloud-based content delivery. It is aided with a utility-based redirection scheme to improve the traffic activities in the world-wide distributed network of MetaCDN.

Declaration

This is to certify that

(i) the thesis comprises only my original work,

(ii) due acknowledgement has been made in the text to all other material used,

(iii) the thesis is less than 100,000 words in length, exclusive of table, maps, bibliographies, appendices and footnotes.

Signature_____

Date_____

Acknowledgments

I would like to express my heartiest gratefulness to God for His divine blessings, which has made it possible to complete this thesis successfully.

Throughout my PhD candidature, I have interacted with a number of researchers whose direct and indirect involvement helped this thesis to come into light. First and foremost, I would like to thank my supervisor Professor Rajkumar Buyya for his guidance, encouragement, views and comments on my research, whilst allowing me the room to work in my own way. With his thoughts and innovative inputs, he always pictures big and provides me a model to emulate. Without his supervision, this thesis would not have been completed. I am grateful to him for providing the opportunity to work in a group of brilliant researchers.

I would like to express gratitude to my PhD committee members Professor Rao Kotagiri and Dr. James Broberg for making incisive comments and constructive suggestions on my research. In particular, it is always fun to discuss research with James. His generous help while working on a practical testbed allowed me to achieve an implementation perspective for my research.

I am thankful to Giancarlo Fortino (University of Calabria, Italy), Christian Vecchiola (University of Melbourne), Marcos Dias de Assunção (INRIA, France), Kris Bubendorfer (Victoria University of Wellington, New Zealand), Athena Vakali (Aristotle University of Thessaloniki, Greece), George Pallis (University of Cyprus), and Kyong Hoon Kim (Gyeongsang National University, South Korea) for their help in my research. On several occasions, they provided excellent advice and insights by reading the early drafts of my papers, and helped to improve the structure, presentation and correctness of those manuscripts.

Many international researchers and industry practitioners helped during my candidature by providing advice and valuable resources such as research papers, technical reports, white papers, and data sheets. Especially, I am grateful to Carlo Mastroianni (ICAR-CNR, Italy), Vivek Pai (Princeton University, USA), Fahim Husain (Akamai Technologies, USA), William Good (Mirror Image Internet, USA), and Lisa Amini (IBM T. J. Watson Research Center, USA).

I would like to acknowledge and thank all the contributing authors of the book, entitled, “Content Delivery Networks”, edited and co-authored by me. I would like to offer special appreciation to Springer and its publishing editor, Christoph Baumann, for helping us to bring out the book in a quick time.

I would like to thank many colleagues and organizations world-wide for their assistance with providing client nodes to conduct experiments on a global testbed. The organizations and associated contacts are: Georgia State University, USA (Sushil Prasad), SUNY Binghamton, USA (Kenneth Chiu), University of California-Irvine, USA (Chris Davison, Nalini Venkatasubramanian), Pennsylvania State University, USA (Konrad Malkowski, Padma Raghavan), University of Rome "Tor Vergata", Italy (Valeria Cardellini), Aristotle University of Thessaloniki, Greece (Athena Vakali, Konstantinos Stamos), University of Innsbruck, Austria (Thomas Fahringer), ICAR-CNR, Italy (Carlo Mastroianni), University of Paris-Sud, France (Paul Malecot), INRIA, France (Gilles Fedak, Eddy Caron,

Laurent Lefevre), Poznan Supercomputing and Networking Center, Poland (Miroslaw Czyrnek), University of Melbourne, Australia (William Voorsluys), Hochschule Furtwangen University, Germany (Christoph Reich, Martin Kramer), Universidad de Castilla-La Mancha, Spain (Agustin Caminero), Universidad Politécnica de Madrid, Spain (Maria Perez), Universidad Complutense de Madrid, Spain (Ignacio Llorente, José Luis Vázquez-Poletti), Autonomous University of Nuevo León, Mexico (Yasmin Rios), Federal University of Campina Grande, Brazil (Francisco Brasileiro), Universidade Federal Fluminense, Brazil (Vinod Rebello), University of Pretoria, South Africa (Serena Coetzee), Gyeongsang National University, South Korea (Kyong Hoon Kim), Saitama University, Japan (Norihiko Yoshida), Osaka Sangyo University, Japan (Akiko Nakaniwa), Beihang University, China (Li Lin), University of Hong Kong (Cho-Li Wang, Roy Ho), and Indian Institute of Science, India (Satish Vadhiyar).

I am thankful to all the past and present members of the CLOUDS Laboratory. In particular, I thank Srikumar Venugopal, Rajiv Ranjan, Jia Yu, Chee Shin Yeo, Anthony Sulistio, Hussein Gibbins, Krishna Nadiminti, Mustafizur Rahman, Marco Netto, Rodrigo Calheiros, SungJin Choi, Charity Lourdes, Saurabh Garg, William Voorsluys, Suraj Pandey, Mohsen Amini, Anton Beloglazov, and Michael Mattess for their help and comments on my work. I also had great time with them outside research. I enjoyed a number of barbeques, picnics, outdoor activities, sightseeing, and sports with them and my other friends from the CSSE department—Jubaer Arif, Andreas Schutt, Adeel Zafar, Archana Sathivelu, Jason Lee, and Lei Ni to name a few. Special thanks to my ex-housemates Monica Chou, Elena Tan, and Kelvin Kong, who pointed out many of my silly mistakes in everyday life, which helped me to improve as a person. I also extend my gratitude to the staff from the CSSE department and School of Engineering, especially, Rosanna Parissi, Dominique Santilli-Centofanti, Binh Phan, Pinoo Bharucha, Julien Reid, and Madalain Dolic for their support during my candidature.

I acknowledge Australian Federal Government, the University of Melbourne, Australian Research Council (ARC), Department of Innovation, Industry, Science, and Research (DIISR), IEEE Technical Committee on Scalable Computing (TCSC), and CLOUDS laboratory for providing scholarships and travel supports to pursue dotoral studies and attend international conferences.

I wish to give heartful thanks to my parents, my brother and sister for their support, dedication and love at all times. Without them it would not be possible to come through the various stages of my life.

Finally, I would like to dedicate this thesis to my fiancée Ziyuan Wang for her warmth and support to enrich my life. I sincerely thank her for the patience and understanding she showed even at times when I had to travel to conferences leaving her all alone to deal with the toughness of the material world!

Al-Mukaddim Khan Pathan
Melbourne, Australia
November 2009

Table of Contents

CHAPTER 1: INTRODUCTION **1**
1.1 Motivation and Scope 2
1.2 Significance of CDN Peering 3
1.3 Research Problem and Objectives 6
1.3.1 Research Issues 6
1.3.2 Objectives 7
1.4 Methodology 7
1.4.1 Implications 8
1.5 Contributions 9
1.6 Thesis Organization 11

CHAPTER 2: CONTENT DELIVERY NETWORKS **13**
2.1 Introduction 13
2.1.1 CDN Components 14
2.2 Background and Related Systems 17
2.2.1 The Evolution of CDNs 17
2.2.2 A Comparative Analysis 20
2.3 Existing CDNs: State of the Art 21
2.3.1 Commercial CDNs 22
2.3.2 Academic CDNs 22
2.3.3 Cloud-based CDNs 23
2.4 A Taxonomy of CDNs 25
2.4.1 Motivations and Scope 26
2.4.2 Taxonomy 28
2.5 Mapping the Taxonomy to CDNs 30
2.5.1 CDN Composition 30
2.5.2 Content Distribution and Management 35
2.5.3 Request-Routing 37
2.5.4 Performance Measurement 39
2.6 Discussion 40
2.7 Positioning the Thesis 41
2.8 Summary and Conclusion 43

CHAPTER 3: CDN PEERING **45**
3.1 Introduction 45
3.2 CDN Peering Initiatives 47
3.3 System Architecture 50
3.3.1 Architectural Components 51
3.3.2 Peering Lifecycle 54
3.3.3 Short-term Resource Negotiation 56

3.3.4 SLA Negotiations 58
3.3.5 Policy Management 59
3.4 Alternative Models for CDN Peering 62
3.4.1 Brokering-based CDN peering 62
3.4.2 QoS-Driven (customized) Brokering-based CDN peering 64
3.5 Challenges to Realize CDN Peering 65
3.6 Technical Issues for CDN Peering 68
3.6.1 Load Distribution for CDN Peering 68
3.6.2 Coordination of CDNs 70
3.6.3 Service and Policy Management 70
3.6.4 Pricing of Content and Services 71
3.7 Summary and Conclusion 71

CHAPTER 4: PERFORMANCE MODELING 73
4.1 Introduction 73
4.1.1 SLAs to Ensure QoS 75
4.2 Performance Models 76
4.2.1 Motivation and Scope 76
4.2.2 Single CDN Model 77
4.2.3 CDN Peering Model 80
4.3 Performance Results 83
4.3.1 QoS Performance of the Primary CDN 84
4.3.2 Request-Redirection Policies 86
4.3.3 Impact of Request-Redirection 89
4.3.4 Measurement Errors 90
4.4 Summary and Conclusion 92

CHAPTER 5: RESOURCE DISCOVERY AND REQUEST-REDIRECTION 93
5.1 Introduction 93
5.2 Motivation and Scope 94
5.2.1 Resource Discovery 97
5.2.2 Server Selection and Request-Redirection 98
5.3 Algorithms 98
5.3.1 Resource Discovery Formulation 98
5.3.2 Request-Redirection Formulation 101
5.3.3 Perceived Benefits 105
5.3.4 Time Complexity 105
5.4 Methodology 106
5.4.1 Simulation Environment and Parameters 106
5.4.2 Schemes and Metrics for Comparison 114
5.5 Experimental Evaluation 115
5.5.1 Traffic Load on Servers 115

5.5.2 Number of Completions 118
5.5.3 Service Disruptions 119
5.5.4 Server Utilization 120
5.5.5 Redirection Performance 121
5.5.6 Sensitivity Analysis 122
5.6 Summary and Conclusion 123

CHAPTER 6: UTILITY OF CDN PEERING 127
6.1 Introduction 127
6.2 Motivation and Scope 128
6.3 Utility Model for CDN Peering 129
6.3.1 Content-Serving Utility 131
6.4 Evaluation Methodology 132
6.4.1 Simulation Environment and Parameters 134
6.4.2 Performance Metrics 135
6.5 Experiment Results 137
6.5.1 Request Traffic vs. Utility 137
6.5.2 Content-Serving Ability 139
6.5.3 Response Time vs. Utility 140
6.5.4 Service Completions 141
6.6 Summary and Conclusion 144

CHAPTER 7: OPTIMIZING UTILITY OF A CONTENT DELIVERY CLOUD 145
7.1 Introduction 145
7.2 Motivation and Scope 146
7.3 The MetaCDN Overlay 148
7.3.1 Overview 148
7.3.2 System Characteristics 149
7.3.3 A Comparative Analysis 151
7.4 Maximizing MetaCDN Utility 156
7.4.1 Problem Formulation 156
7.4.2 Quantitative Expressions 157
7.5 Request-Redirection Design 158
7.5.1 Design Space 158
7.5.2 Utility-based Request-Redirection 160
7.6 Evaluation Methodology 162
7.6.1 Schemes and Metrics for Comparison 164
7.7 Empirical Results 166
7.7.1 Response Time Observations 166
7.7.2 Throughput Observations 170
7.7.3 MetaCDN Utility 171
7.7.4 Content Provider's Perceived Utility 172

7.8 Summary and Conclusion 173

CHAPTER 8: CONCLUSION AND FUTURE DIRECTIONS 175
8.1 Summary 175
8.2 Future Directions 179
8.2.1 Content Replication in Cooperative Domain 179
8.2.2 Active Content 180
8.2.3 On the Economics of Cooperation 180
8.2.4 Scalability of Dynamic Content Delivery 181
8.2.5 Integration of Media Streaming 182
8.2.6 Mobility for Content Delivery 182
8.2.7 Applicability of Anycasting 182
8.2.8 Energy-Aware Content Delivery 183
8.2.9 Enhancement for Content Delivery Clouds 183

REFERENCES 185

List of Figures

Figure 1.1: Abstract view of a CDN. 2
Figure 1.2: A CDN peering scenario. 4
Figure 1.3: Abstraction of CDN peering. 8
Figure 1.4: Thesis contributions and organization. 9

Figure 2.1: Architectural components of a CDN. 14
Figure 2.2: Content/services provided by a CDN. 16
Figure 2.3: Evolution of CDN technologies. 19
Figure 2.4: Request-routing in a CDN. 29

Figure 3.1: Example of CDN peering. 46
Figure 3.2: Architecture of a system to assist the creation of CDN peering. 50
Figure 3.3: Interaction flows in a CDN peering arrangement. 53
Figure 3.4: Flowchart for a short-term resource negotiation through auction. 58
Figure 3.5: Basic policy framework. 60
Figure 3.6: Brokering-based approach to form CDN peering. 63
Figure 3.7: QoS-driven brokering-based approach for CDN peering. 64

Figure 4.1: A CDN modeled as an M/G/1 queue. 77
Figure 4.2: Cumulative distribution of waiting time. 80
Figure 4.3: Conceptual view of CDN peering. 81
Figure 4.4: A peering scenario with three CDNs. 83
Figure 4.5: Cumulative distribution of waiting time (primary) without peering. 85
Figure 4.6: Cumulative distribution of waiting time (primary) in peering. 85
Figure 4.7: Redirection ratio at different primary CDN loads. 87
Figure 4.8: Distribution of redirected requests to peers. 88
Figure 4.9: Impact of request-redirection on waiting time of the primary CDN. 89
Figure 4.10: Impact of measurement errors on redirection ratio. 91
Figure 4.11: Waiting time variation for different load measurement error. 91

Figure 5.1: Service Request from the primary CDN during resource discovery. 99
Figure 5.2: Service response from a peer's server during resource discovery. 99
Figure 5.3: Load distribution algorithm (*LD_minCost*). 104
Figure 5.4: Reference Scenario for simulation. 107
Figure 5.5: Operations of the mediator. 111
Figure 5.6: Network proximity measurement 112
Figure 5.7: Number of concurrent requests for each scheme 116
Figure 5.8: Number of completed requests at each server in different schemes. 118
Figure 5.9: Total completions in each scheme. 118
Figure 5.10: Service rejection rate for each scheme. 119
Figure 5.11: Server utilization for each scheme. 120
Figure 5.12: Average utilization of the primary for each scheme. 121

Figure 5.13: Comparison of the request-redirection schemes. 122
Figure 5.14: Sensitivity to utilization. 123
Figure 5.15: Sensitivity to traffic distribution. 123

Figure 6.1: The CDN peering system as a content-utility system. 130
Figure 6.2: Major operational steps in the content-utility system. 130
Figure 6.3: A schematic representation of the evaluation methodology. 133
Figure 6.4: Utility measures for different traffic types. 137
Figure 6.5: Cumulative frequency of minimum utility. 139
Figure 6.6: Sensitivity to traffic distribution. 140
Figure 6.7: Relation between mean response time and content serving utility. 141
Figure 6.8: Number of completions in each CDN server. 142
Figure 6.9: Total completions in each CDN and in the CDN peering system. 142
Figure 6.10: Service rejection rate in the CDN peering system. 143
Figure 6.11: Total completions and utility for CDN peering. 143

Figure 7.1: Components of the MetaCDN overlay system. 148
Figure 7.2: Utility-based request-redirection in MetaCDN. 160
Figure 7.3: Experiment testbed. 163
Figure 7.4: Response time obtained in each client location. 168
Figure 7.5: Average throughput obtained in each client location. 169
Figure 7.6: MetaCDN utility over time. 171
Figure 7.7: Probability of achieving specified utility 172

List of Tables

Table 2.1: Comparison between CDNs and related systems. 21
Table 2.2: Summary of representative commercial CDNs. 22
Table 2.3: Summary of existing academic CDNs. 23
Table 2.4: Feature comparison of cloud-based CDNs. 24
Table 2.5: CDN taxonomy. 28
Table 2.6: CDN composition taxonomy mapping. 32
Table 2.7: Content distribution and management taxonomy mapping. 34
Table 2.8: Request-routing taxonomy mapping. 36
Table 2.9: Performance measurement taxonomy mapping. 38

Table 3.1: List of commonly used terms. 51
Table 3.2: Policy components mapping. 61
Table 3.3: Comparison of CDN models. 62

Table 4.1: Parameter and expressions for the analytical model. 77
Table 4.2: List of notations used in the CDN peering scenario. 84
Table 4.3: Workload model. 84
Table 4.4: Reduction of waiting time on the primary CDN. 90

Table 5.1: Annotation of the resource discovery algorithm. 101
Table 5.2: Significant properties of the request-redirection scheme. 103
Table 5.3: Parameters for the system model. 108
Table 5.4: Configuration of the simulated CDN Web servers. 109
Table 5.5: List of Performance indices. 114

Table 6.1: Parameters used for evaluation. 134
Table 6.2: List of performance indices. 136

Table 7.1: Summary of the experiment. 163
Table 7.2: List of performance indices. 166
Table 7.3: Content providers' benefits on user perceived performance. 173

Chapter 1

Introduction

Over the last decades, Web users have witnessed the spread and maturity of the Internet that have caused enormous growth in network traffic, driven by the rapid acceptance of broadband access, increase in systems complexity, and content richness. The ever-evolving nature of the Internet brings new challenges in managing and delivering content to end-users. For example, popular Web services often suffer congestion and bottlenecks due to the large demands posed on them. Such a sudden spike in Web content requests (e.g. the one occurred during the 9/11 terrorist attack in the USA) is often termed as flash crowds [14] or SlashDot [2] effects. It may cause heavy workload on particular Web server(s), and as a result a hotspot [14] can be generated. Coping with such unexpected huge request traffic causes significant strain on a Web server and eventually the Web servers are totally overwhelmed with the sudden increase in traffic. This causes the Web site holding the content to become temporarily unavailable.

Content Delivery Networks (CDNs) [33, 98, 152, 166, 176, 211, 215] have emerged to overcome these limitations by offering infrastructure and mechanisms to deliver content and services in a scalable manner, enhancing end-users Web access experience, in terms of response time and system throughput. A CDN (Figure 1.1) is a collaborative collection of network elements spanning the Internet, where content is replicated over several mirrored Web servers (i.e. surrogate) located at the *edge* of the network to which end-users are connected. Content is replicated by the CDN's content distributor, either on-demand or beforehand, by pushing the content from the origin server to the surrogates. As an end-user is served with the desired content

from the nearby server, the user ends up unknowingly communicating with a replicated surrogate close to him/her and retrieves content from that server.

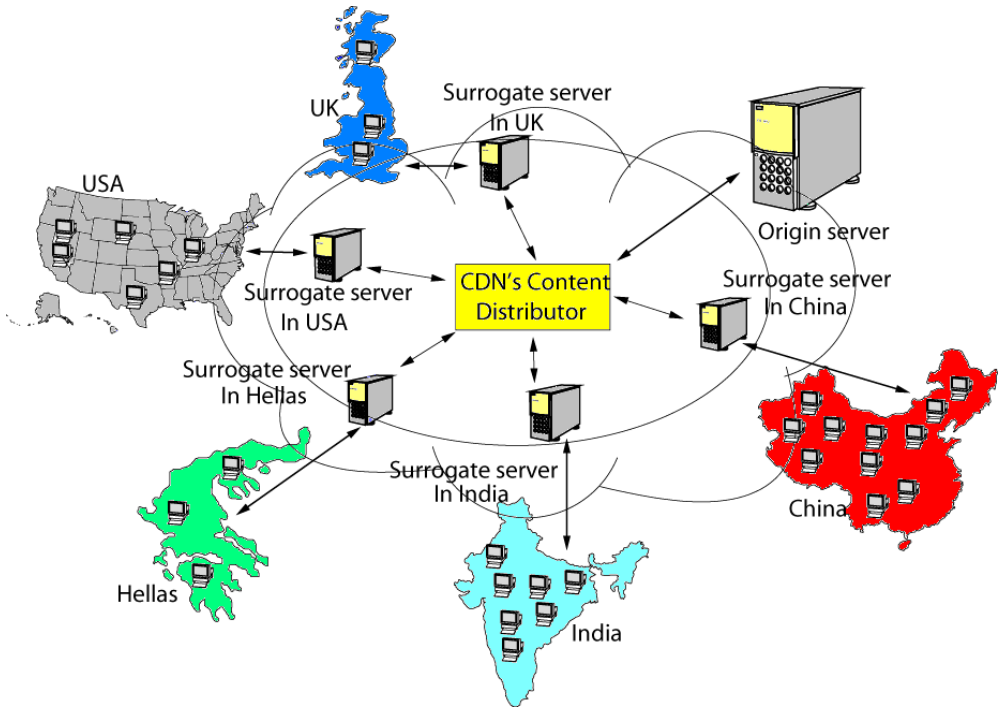


Figure 1.1: Abstract view of a CDN.

1.1 Motivation and Scope

Running a global CDN is challenging in technical, administrative, and financial terms. A CDN provider places Web server clusters at numerous geographical locations worldwide to deliver content on behalf of content providers under specific Service Level Agreements (SLAs) [26]. These SLAs detail commitments for the CDN provider to deliver services within certain response time and throughput constraints. A CDN provider realizes monetary penalization if it fails to meet SLA-bound commitments to provide high quality service to end-users. The provision for such requirements forms a substantial entry barrier for new CDNs, as well as affects the commercial viability of existing ones. This is evident from the major consolidations of the CDN market, down to a handful of key players, which has occurred in recent years [5].

To operate effectively a CDN is required to either over-provision its resources or harness external resources on demand. Unfortunately, due to the proprietary nature,

existing CDNs do not cooperate in delivering content to end-users. A content provider typically subscribes to one CDN and thus cannot use the resources of multiple CDNs at the same time. Such a closed, non-cooperative model results in disparate CDNs who are often prone to resource over-provisioning, degraded performance, SLA violations, and high operational costs. Internetworking of distinct CDNs is one of the possible solutions to handle flash crowds [14] and anticipated increases in demand, Web resources over-provisioning, and adverse business impact by providing cooperative content delivery among CDNs. It can also achieve economics of scale, in terms of cost effectiveness and performance for both providers and end-users.

Analysis of previous research efforts [10, 24, 72, 79, 119, 208, 210] in this context reveals that there has been only modest progress on the frameworks and policies required to allow internetworking between providers. The reasons for this lack of progress are due to the complexity of the technological problems, legal, and commercial operational issues that need to be solved in the practical context.

This thesis sets out to move beyond the shortcomings of prevalent commercial CDNs that limit the scope and scale of their services. We term the technology for interconnection and interoperation between CDNs as "peering" and the resulting system as "CDN peering", as defined in the following:

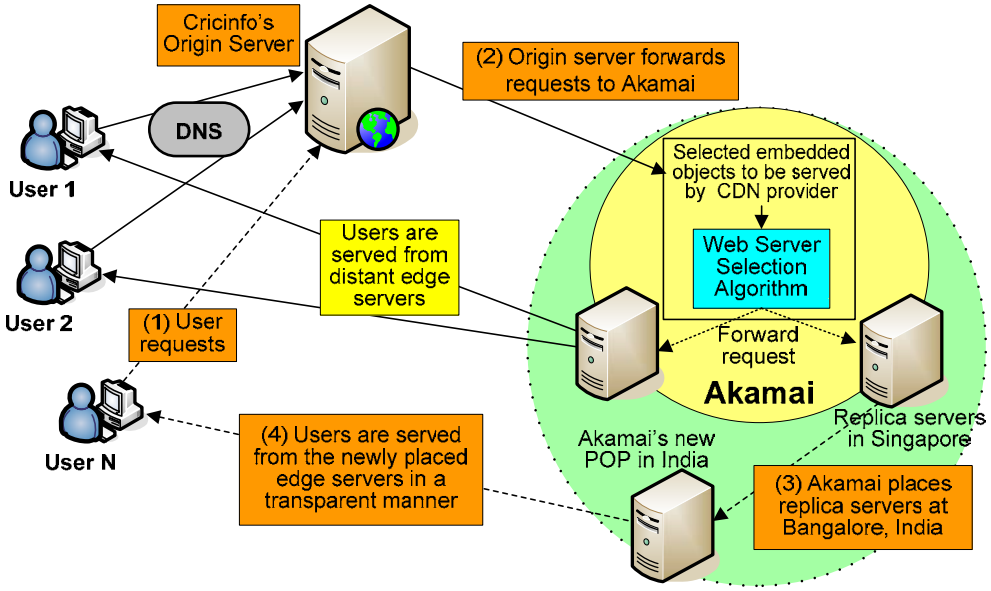
> *CDN peering is formed by a set of autonomous CDNs {CDN_1, CDN_2, …, CDN_n}, which cooperate through an interconnection mechanism **M** that provides facilities and infrastructure for sharing resources in order to ensure efficient service delivery. Each CDN_i is connected to other peers through a 'conduit' C_i that assists in discovering external resources that can be harnessed from other CDNs.*

The rest of this chapter establishes the significance and relevance of our proposal by describing specific research issues, objectives, solution methodology, and implications, followed by thesis contributions and organization.

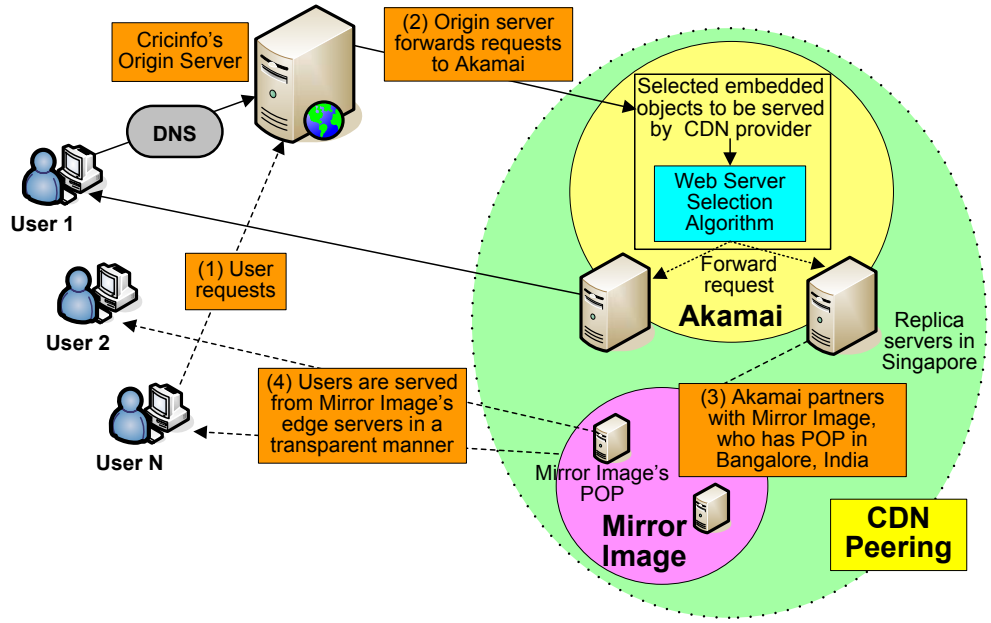
1.2 Significance of CDN Peering

Flash crowds [14] occur due to external events of extreme magnitude and interest or sudden increases in visibility after being linked from popular high traffic Websites,

such as Slashdot or Digg. As a consequence, Web sites often suffer congestions, bottlenecks, and lengthy downtime due to large demands made on the resources of the provider hosting them. Increases in Web traffic can also occur due to the staging of popular events, such as the Olympic Games or the FIFA World Cup.



(a) Expansion of a CDN provider through placing new POP.



(b) Internetworking between CDN providers.

Figure 1.2: A CDN peering scenario.

Historical evidences show that the level of demand generated for popular Websites, e.g. 12 million requests per hour to the 1998 Soccer World Cup Website [14] and 56.8 million requests on a peak day to the 1998 Winter Olympic Website [101], can be impossible to satisfy using a single server, or even a cluster. During September 11, 2001, server availability approached 0% for many popular news Websites with pages taking over 45 seconds to load, if at all [148]. Given that end-users will wait as little as 10 seconds before aborting their requests, this can lead to further bandwidth and resource wastage [96]. More recently, the Internet community has observed instances of flash crowds, such as the live inaugural address of US President Barack Obama in 2009, producing 54% spike in global Internet traffic [89]. Akamai and Limelight Networks have been reported to respectively deliver more than 7 million simultaneous video streams [6] and 2.5 million streams with 250 Gbps data [127].

CDN peering, as a solution to handle flash crowds, allows providers to cooperatively achieve greater scale and reach, as well as service quality and performance, than they could otherwise attain individually. Its significance can be better understood by the scenario in Figure 1.2. The ICC Cricket World Cup 2011 is being held in South Asia, and *www.cricinfo.com* is providing the live media coverage. As a content provider, *www.cricinfo.com* has an exclusive SLA with the CDN provider, Akamai [76]. However, Akamai does not have enough Point of Presence (POP) in India, Sri Lanka and Bangladesh, where the event will be held. Given that cricket is the most popular sport in South Asia, people of the hosting countries will have enormous interest in the live coverages provided by *www.cricinfo.com*. Since Akamai will be aware of this event well in advance, its management can take steps to deal with the evolving situation. To better serve end-users, Akamai may decide to extend its reach by placing more surrogates in South Asia, or they may use distant edge servers in Singapore (Figure 1.2(a)). Nevertheless, placing new surrogates just for one event will be costly and may not be useful after the event. Akamai risks its reputation if it can not provide the agreed Quality of Service (QoS) to user requests, thus violating SLAs and causing profit reduction. Therefore, the solution for Akamai is to cooperate with other CDNs to deliver the service that it cannot provide otherwise (Figure 1.2(b)).

Such resource sharing and cooperation between CDNs may vary in terms of the purpose, scope, incoming traffic size, and duration. In the face of flash crowds, CDN peering should be automated to react within a tight time frame—as it is unlikely that a human directed negotiation will occur quickly enough to satisfy the evolved niche. For long-duration events (Figure 1.2), a human-directed agent can negotiate to ensure that the resulting decisions comply with the strategic goals of participating CDNs.

1.3 Research Problem and Objectives

This thesis tackles the research challenges in relation to the development of advanced, high performance, and cost effective content delivery solutions by enabling coordination and cooperation between multiple content delivery services.

An ad-hoc or planned peering of CDNs requires fundamental research to be undertaken to address the core problems of load index measurement and dissemination; request assignment and redirection; content replication; and sustained resource sharing among participants on a geographically distributed Internet-scale. These issues are deeply interrelated and co-dependent for a single CDN. However, they must now be considered in a cooperative manner among peered CDNs, whilst satisfying the complex multi-dimensional constraints placed on each individual provider.

1.3.1 Research Issues

While CDN peering is appealing, the challenges in adopting it include architecting a system that virtualizes multiple providers and offloads excessive user requests, through distributed request-redirection, from overloaded CDN servers to least loaded servers based on cost, performance, and incoming traffic. In particular, we identify and investigate the following five key research issues:

- ***When to peer.*** The circumstances that trigger peering. The initiating condition must consider expected and unexpected load increases.
- ***How to peer.*** The strategy taken to motivate and form CDN peering. Such a strategy must specify the interactions, accounting, and revenue-sharing among entities and allow the usage of divergent policies for operations.

- ***Whom to peer with.*** The decision making mechanism used for choosing CDNs to peer. It includes predicting performance of providers, working around issues of separate administration and limited information sharing between multiple CDNs.
- ***How to manage and enforce operational policies.*** How operational policies are managed to ensure effective cooperation. It includes deploying necessary request-redirection and load distribution strategies and administering them.
- ***How to demonstrate peering applicability.*** The process to show the usefulness of explored strategies, which relates to their *proof-of-concept* implementation.

1.3.2 Objectives

To deal with the challenges associated with the above research issues, we delineate the following objectives:

1. Position the concept of CDN peering by providing a systematic understanding and categorization of the CDN space in terms of applications, features, and implementation techniques.
2. Provide a novel architecture for CDN peering to enable resource sharing among multiple CDNs and demonstrate the effectivenss of CDN peering.
3. Investigate enabling techniques, i.e. resource discovery, server selection, and request-redirection, in relation to CDN peering.
4. Investigate the QoS-oriented aspect of request servicing via a utility model for CDN peering and demonstrate its applicability in practice.

1.4 Methodology

A model for CDN peering takes the form of an overlay network consisting of resources spanning multiple providers, as depicted in Figure 1.3. Such an arrangement serves as a means to address unexpected spike in incoming traffic, as well as anticipated short or long-term increases in demand, when a single CDN has insufficient resources. From the figure we see that end-user requests for content are made to the Request Routing System (RRS) of a CDN provider. The RRS is equipped with neces-

sary request-redirection and load distribution mechanisms to forward user requests either directly to its server(s) or to a peer. As for instance, some user requests of CDN 1 are served by its local server or the origin server (on cache miss), whereas others are being served by external Web servers of a peer, CDN 2. Again in the same peering relationship, when another CDN experiences high load (for example, CDN 2), its users may be served by its peer's (CDN N) servers over this shared infrastructure.

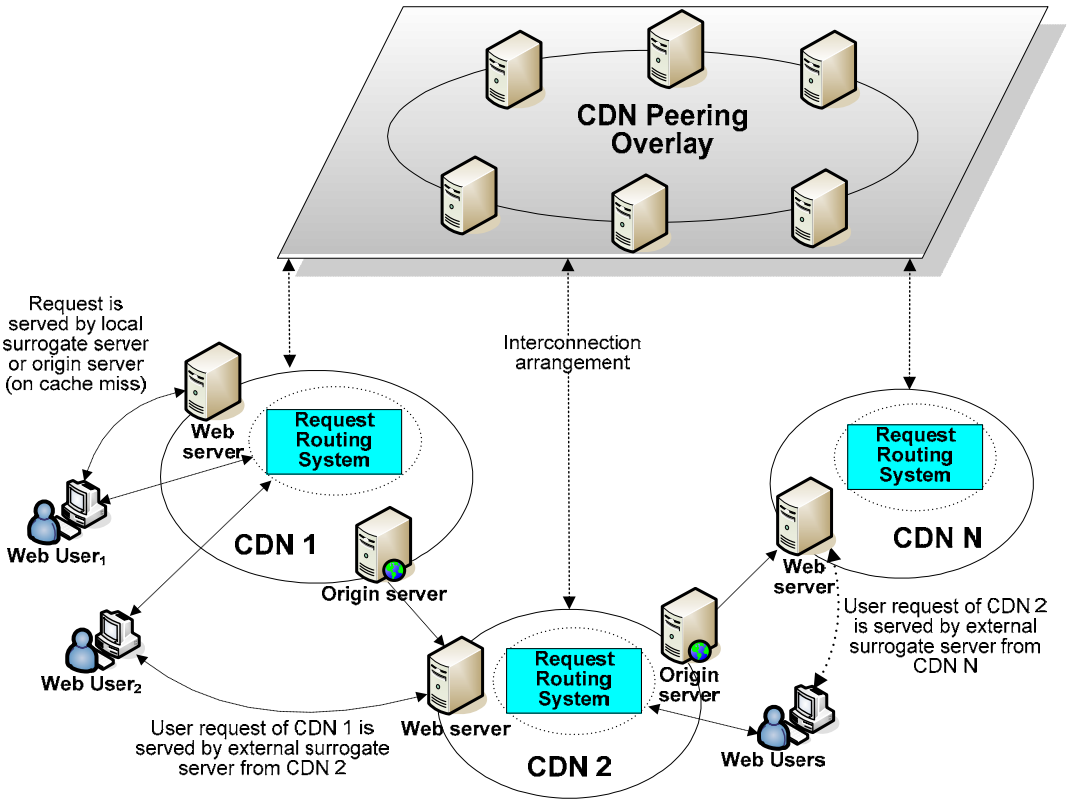


Figure 1.3: Abstraction of CDN peering.

1.4.1 Implications

With the aid of the methodology described above, we address the research issues stated in Section 1.3.1. Specifically, an architectural model is proposed to answer when and how to initiate peering. It is supported by an analytical model to demonstrate peering effectiveness. Novel techniques are proposed to investigate how external resources of disparate CDNs are discovered and how the optimal server is selected for distributed load sharing. The proposed mechanisms are evaluated

through discrete-event simulations by conducting *repeatable* experiments in a *controlled* environment. The notion of CDN peering is realized through empirical studies on the MetaCDN system [29, 157], which is an integrated overlay network to enable high performance content delivery.

1.5 Contributions

The contributions of the thesis broadly fall into three categories: *architectural models*, *peering mechanisms* and *implementation*. Figure 1.4 shows this categorization.

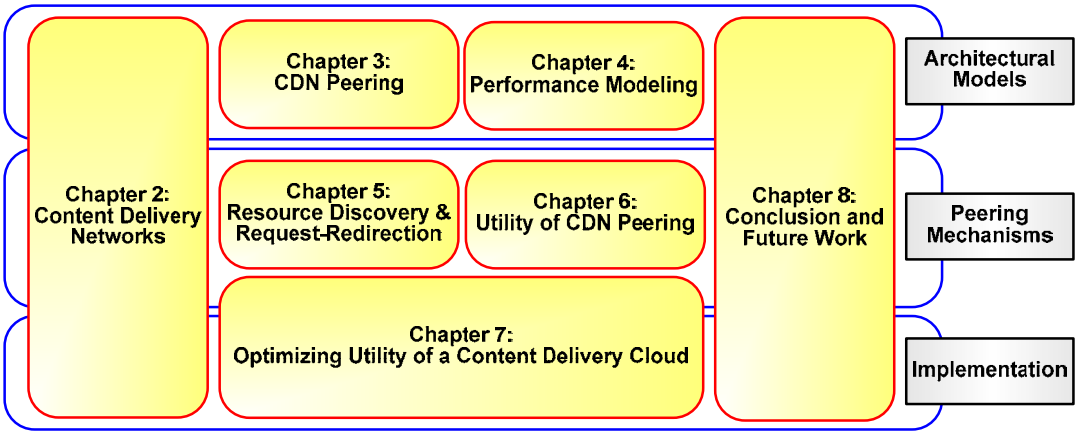


Figure 1.4: Thesis contributions and organization.

In the following we detail the major thesis contributions:

1. ***CDN taxonomy and insights.*** This thesis presents a taxonomy of CDNs based on the state of the art. It investigates related key concepts and technologies, describes the background highlighting the evolution of CDNs, and identifies uniqueness from other related distributed computing paradigms. The presented taxonomy is mapped to present-day CDNs to demonstrate its applicability to categorize and analyze the content delivery landscape. Thus, this thesis provides a roadmap for researchers to interpret related essential concepts and assists to perform a "gap" analysis in this domain.
2. ***CDN peering architecture.*** This thesis introduces an architectural model for CDN peering. It describes the components, architectural features, use cases, and formation of peering arrangements. In addition, it presents two comple-

mentary models to assist CDN peering, compares the proposed models, and summarizes their unique features. The challenges and core technical issues to implement CDN peering are also discussed, thus establishing the basis to develop necessary enabling techniques.

3. ***Analytical models.*** This thesis presents analytical models for CDN peering based on the fundamentals of queuing theory. Along with the derivation of the performance models, it conducts a systematic analysis to study the impact of key performance parameters such as load and measurement errors that can be expected from a real system. The presented models assist to demonstrate the effects of CDN peering and predict end-user perceived performance. It also facilitates in making concrete QoS guarantee for a CDN provider.
4. ***Enabling techniques for CDN peering.*** This thesis investigates the techniques to enable CDN peering. For this purpose, it presents resource discovery and request-redirection mechanisms, coupled with an optimal server selection strategy to perform load sharing in a CDN peering arrangement. The proposed mechanisms are evaluated through discrete-event simulations and a sensitivity analysis is performed using critical system parameters. Thus, this thesis demonstrates the novelty and effectiveness of the proposed techniques to achieve service "responsiveness" even under degenerated load conditions.
5. ***A utility model.*** This thesis introduces a utility model for CDN peering to measure its content-serving ability. The utility model captures the traffic activities in the system and helps to reveal the true propensities of participating CDNs to cooperate in peering. Through extensive simulations, this thesis unveils interesting observations on how the utility is varied for different system parameters and provide incentives for their exploitation in the system design.
6. ***Proof-of-concept implementation.*** This thesis provides an implementation perspective for CDN peering, realized through the MetaCDN overlay system [29, 157]. It investigates the design space of a competent request-redirection technique for MetaCDN and reports a *proof-of-concept* empirical study on a global testbed to evaluate the performance of a utility-based re-

quest-redirection technique and provide insights on the MetaCDN utility. Thus, this thesis addresses the problem of designing a redirection technique in practical context that is both responsive to a wide range of workload variations, and uplift end-user perceived performance.

To summarize, the work presented in this thesis is in line with the current trends in CDNs to shift towards a utility computing model [39], which allows customers to exploit content delivery services without having to build a dedicated infrastructure [92, 207]. Therefore, it is our thesis to present advanced and efficient content delivery solutions that are scalable, high performance and cost effective.

1.6 Thesis Organization

The core chapters of this thesis are derived from various papers published during the PhD candidature. As shown in Figure 1.4, the remainder of the thesis is organized as:

- ***Chapter 2: Content Delivery Networks.*** This chapter provides an in-depth analysis and overview of CDN technologies. It presents a taxonomy of CDNs with a complete coverage of the field. This chapter is partially derived from:
  - **A.-M. K. Pathan**, R. Buyya, and A. Vakali, "CDNs: State of the art, insights, and imperatives," *Content Delivery Networks*, R. Buyya et al. *eds.*, pp. 3-32: Springer-Verlag, Germany, 2008.
  - **A.-M. K. Pathan** and R. Buyya, "A taxonomy of CDNs," *Content Delivery Networks*, R. Buyya et al. *eds.*, pp. 33-77: Springer-Verlag, Germany, 2008.
- ***Chapter 3: CDN Peering.*** This chapter presents an architecture to enable CDN peering. It describes the key concepts and identify the associated challenges to realize CDN peering. This chapter is partially derived from:
  - R. Buyya, **A.-M. K. Pathan**, J. Broberg, and Z. Tari, "A Case for Peering of Content Delivery Networks," *IEEE DSOnline*, vol. 7, no. 10, pp. 3, 2006.
  - **A.-M. K. Pathan**, J. Broberg, and R. Buyya, "Internetworking of CDNs," *Content Delivery Networks*, R. Buyya et al. *eds.*, pp. 389-413: Springer-Verlag, Germany, 2008.

- ***Chapter 4: Performance Modeling.*** This chapter develops performance models to highlight on peering effectiveness. The analytical models presented in this chapter are derived from the following publication:
  - **A.-M. K. Pathan** and R. Buyya, "Architecture and performance models for QoS-driven effective peering of content delivery networks," *Multiagent and Grid Systems Journal,* vol. 5, no. 2, pp. 165-195, 2009.
- ***Chapter 5: Resource Discovery and Request-Redirection.*** This chapter presents enabling technologies for CDN peering so as to perform load sharing and handle flash crowds. The mechanisms and the simulation results presented in this chapter are derived from the following publication:
  - **A.-M. K. Pathan** and R. Buyya, "Resource discovery and request-redirection for dynamic load sharing in multi-provider peering content delivery networks," *Journal of Network and Computer Applications*, vol. 32, no. 5, pp. 976-990, 2009.
- ***Chapter 6: Utility of CDN Peering.*** This chapter lays out a utility model for CDN peering. It describes the impact of key system parameters on utility. This chapter is derived from the following publication:
  - **A.-M. K. Pathan** and R. Buyya, "A utility model for peering of multi-provider content delivery services," *Proc. 34th IEEE International Conference on Local Computer Networks (LCN'09)*, IEEE CS Press, USA, pp. 475-482, 2009.
- ***Chapter 7: Optimizing Utility of a Content Delivery Cloud.*** This chapter presents the results of a measurement study on the global MetaCDN testbed. It is partially derived from the following publication:
  - **A.-M. K. Pathan**, J. Broberg, and R. Buyya, "Maximizing utility for content delivery clouds," *Lecture Notes in Computer Science, Proc. 10th International Conference on Web Information Systems Engineering (WISE'09)*, LNCS 5802, Springer, Germany, pp. 13-28, 2009.
- ***Chapter 8: Conclusion and Future Directions.*** The concluding chapter provides a summary of contributions and a comprehensive future research roadmap.

Chapter 2

Content Delivery Networks

Content Delivery Networks (CDNs) is a thriving research field, where advanced technologies, solutions and capabilities are being introduced constantly. This chapter provides an in-depth analysis of CDN technologies, origins, and evolution by capturing a "snapshot" of the state of the art in this domain. This chapter also presents a taxonomy of CDNs with a complete coverage of this field to provide a comprehensive account of applications, features, and implementation techniques. The taxonomy provides a basis for categorizing related solutions and systems. It is mapped to a few representative systems to validate its accuracy and demonstrate its applicability to present-day CDNs. This mapping assists to perform a "gap" analysis in this domain by interpreting related concepts.

2.1 Introduction

In the context of CDNs, *content delivery* describes an action of servicing content based on end-user requests. *Content* refers to any digital data resources and it consists of two main parts: the *encoded media* and *metadata* [171]. The encoded media includes static, dynamic, and continuous media data (e.g. audio, video, documents, images and Web pages). Metadata is the content description that allows identification, discovery, and management of multimedia data, and facilitates its interpretation. Content can be pre-recorded or retrieved from live sources; it can be persistent or transient data within the system [171]. A CDN can be seen as a new virtual overlay to the Open Systems Interconnection (OSI) network reference model [100]. It provides overlay network services relying on application layer protocols such as Hyper Text Transfer Protocol (HTTP) or Real Time Streaming Protocol (RTSP) for transport [72].

The three main entities in a CDN system are the following: *content provider*, *CDN provider*, and *end-users*. A *content provider* or *customer* is the one who delegates the Uniform Resource Identifier (URI) name space of the Web objects to be distributed. The *origin server* of the content provider holds those objects. A *CDN provider* is a proprietary organization or a company that provides infrastructure facilities to content providers in order to deliver content in a timely and reliable manner. *End-users* or *clients* are entities who access content from the content provider's Web site.

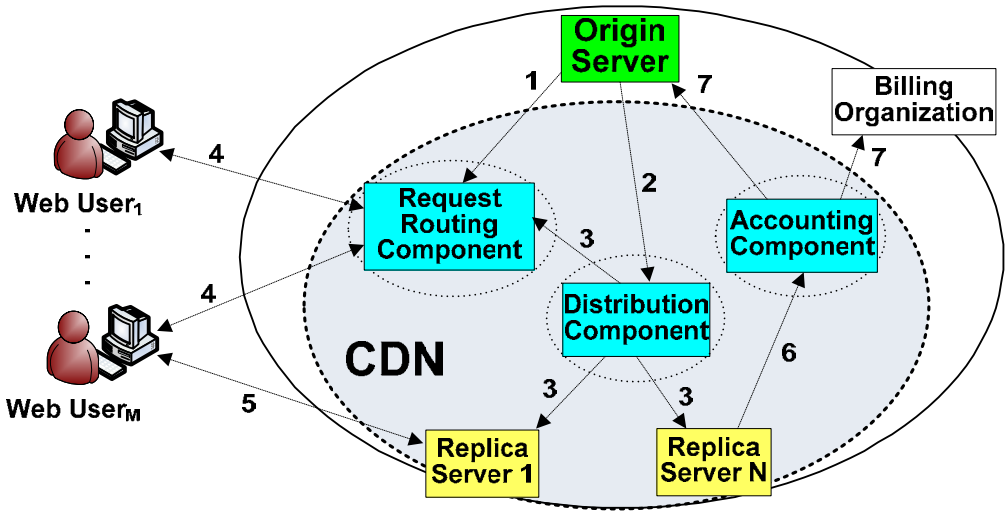


Figure 2.1: Architectural components of a CDN.

CDN providers use *caching* and/or *replica servers* located in different geographical locations to replicate content. CDN cache servers are also called *edge servers* or *surrogates*. A collective set of CDN edge servers are often referred to as a *Web cluster*. CDNs distribute content to edge servers in such a way that all of them share the same content and URI. End-user requests are redirected to the nearby optimal edge server and it delivers requested content to end-users. Thus, transparency for users is achieved. Additionally, edge servers populate the accounting information on the delivered content for traffic reporting and billing purposes.

2.1.1 CDN Components

Figure 2.1 shows the architecture of a CDN system comprising four main components. These components and their operational steps are described in the following:

- ***Content-delivery component.*** It consists of the origin infrastructure (application severs, Web servers and databases hosted by content provider) and edge infrastructure (a set of replica servers hosted by the CDN). The origin server delegates the URI namespace of the content to the request-routing component (step 1) and publishes the content into the distribution component (step 2).
- ***Distribution component.*** It moves content from the origin server to the replica servers and ensures consistency of the content in the caches. It also providees feedback to the request-routing component in order to assist in replica server selection for delivering content to end-users (step 3).
- ***Request-routing component.*** It receives end-user content requests due to the URI namespace delegation and directs them to appropriate replica servers (step 4). It also interacts with the distribution component to keep an up-to-date view of the content stored in the replica servers. The selected replica server delivers the requested content to end-users (step 5) and sends accounting information of the delivered content to the accounting component (step 6).
- ***Accounting component.*** It maintains logs of user accesses and records the usage statistics of CDN resources. This aggregated information is used for traffic reporting and usage-based billing by the origin server or a third-party billing organization (step 7). The recorded statistics are also used as feedback to the request-routing component.

With the aid of the above components, a CDN provider focuses on providing the following services and functionalities: storage and management of content; distribution of content among edge servers; cache management; delivery of static, dynamic, and streaming content; backup and disaster recovery solutions; monitoring services; performance measurement; and reporting.

A content provider (i.e. customer) signs up with a CDN for service and has its content placed on the replica servers. In practice, CDNs typically host third-party content including static content (e.g. static HTML pages, images, documents, software patches), streaming media (e.g. audio, real time video), User Generated Videos (UGV), and varying content services (e.g. directory service, e-commerce service, file

transfer service). The sources of content include large enterprises, Web service providers, media companies, social networking Web sites and news broadcasters. Typical customers of a CDN are media and Internet advertisement companies, data centers, Internet Service Providers (ISPs), online music retailers, mobile operators, consumer electronics manufacturers, and other carrier companies. Each of these customers wants to publish and deliver its content to end-users on the Internet in a reliable and timely manner. End-users can interact with the CDN by specifying the content/service requests through cell phones, smart phones/PDAs, laptops or desktops. Figure 2.2 depicts the different content/services served by a CDN to end-users.

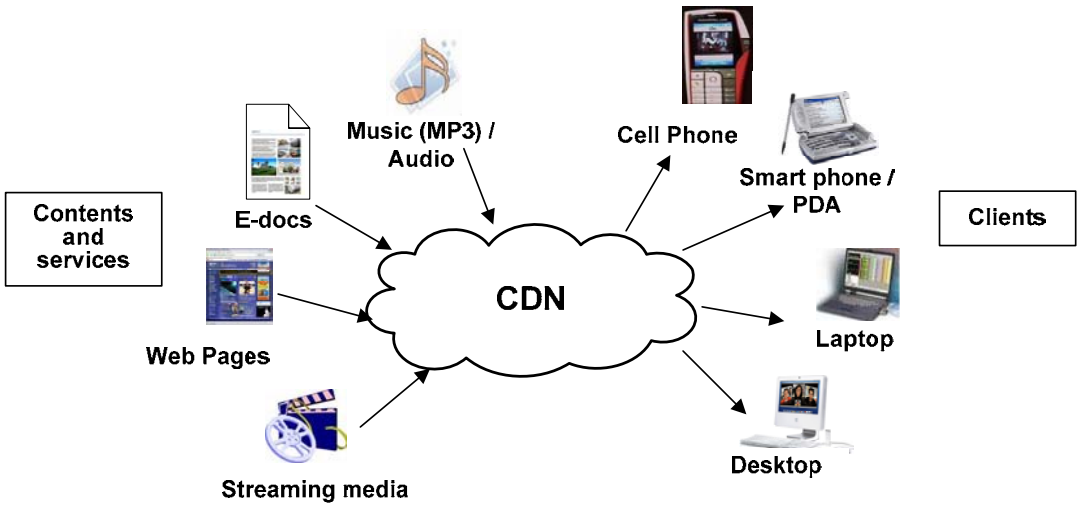


Figure 2.2: Content/services provided by a CDN.

CDN providers charge their customers according to the content delivered (i.e. traffic) to end-users by their replica servers. The average price of CDN services is quite high [186], often out of reach for many Small-to-Medium Enterprises (SME) or not-for-profit organizations. The price of CDN services are mainly affected by [152],

- *Bandwidth usage*, which is measured by the CDN provider to charge (per Mbps) customers typically on a monthly basis;
- *Variation of traffic distribution*, which characterizes pricing under different situations of congestion and bursty traffic;
- *Size of the content replicated over edge servers*, which is a critical criterion for posing charges (e.g. price per GB) on customer audiences;

- *Number of edge servers*, which captures the ability of a CDN to offer content at charges that will not exceed the charges in typical caching scenarios; and
- *System reliability*, *stability* and *security* issues for outsourcing content delivery. It also inhibits a cost of sharing confidential data, which varies over different content providers on the basis of the protected content type.

2.2 Background and Related Systems

Content providers earn significant financial incentives from Web-based e-business and they are concerned on improving the service quality experienced by end-users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructure supporting these technologies forms a new type of network, which is often referred to as a 'content network' [171].

2.2.1 The Evolution of CDNs

Several content networks attempt to address the performance problem by using different mechanisms to improve the Quality of Service (QoS) experienced by end-users:

- ***Improved Web server.*** An initial approach is to modify the traditional Web architecture by improving the server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not scalable, since small enhancements are not possible and at some point, the complete server system might have to be replaced [98].
- ***Caching proxy deployment by an ISP.*** It can be beneficial for the narrow bandwidth users accessing the Internet, since to improve performance and reduce bandwidth utilization, caching proxies are deployed close to end-users. Caching proxies may also be equipped with technologies to detect a server failure and maximize efficient use of caching proxy resources. Users often configure their browsers to send Web requests through these caches rather than sending directly to origin servers. When this configuration is properly done, a user's entire browsing session goes through a specific proxy. Thus, the caches

contain most popular content viewed by all the users of the caching proxies.

- ***Hierarchical caching and server farms.*** A provider may also deploy different levels of local, regional, international caches at geographically distributed locations. Such arrangement is referred to as *hierarchical caching*. This may provide additional performance improvements and bandwidth savings [22]. The establishment of server farms is a more scalable solution that has been in widespread use for several years. A server farm comprises multiple Web servers, each of them sharing the burden of answering requests for the same Web site [98]. It also makes use of a Layer 4-7 switch (intelligent switching based on information such as URL requested, content type, and username, which can be found in layers 4-7 of the OSI protocol stack of the request packet), also called Web switch or content switch that examines content requests and dispatches them among the group of servers. A server farm can also be constructed with surrogates instead of a switch [67]. This approach is more flexible and shows better scalability. Moreover, it provides the inherent benefit of fault tolerance. Deployment and growth of server farms progress with the upgrade of network links that connect Web sites to the Internet.
- ***CDNs.*** Traditional hosting solutions such as server farms and hierarchical caching proxies fall significantly short of meeting critical requirements of availability, performance and scalability [180]. Some of these shortcomings can be alleviated by *multihoming* (multiple ISPs connectivity to an origin site) and *mirroring* (multiple origin sites). However, Internet middle-mile degradations and scalability remain issues. Furthermore, network and server resources need to be over-provisioned during failures, since a subset of links and/or data centers must be able to handle the entire traffic load for a content site. To address these limitations, *Content Delivery Network* has been deployed in the late 1990s.

With the introduction of CDNs, content providers started putting their Web site content on a CDN. Soon they realized its usefulness through receiving increased reliability and scalability without the need to maintain expensive infrastructure. Hence, several initiatives kicked off for developing infrastructure for CDNs. As a conse-

quence, Akamai Technologies [76] evolved out of an MIT research effort aimed at solving the flash crowd problem. Scientists developed a set of breakthrough algorithms for routing and replicating content over a large network of distributed servers spanning the globe. Within a couple of years, several companies became specialists in providing fast and reliable delivery of content, and CDNs became a huge market for generating large revenues. The flash crowd events [14, 105], such as during the 9/11 terrorist attack in the USA, resulted in serious caching problems for some sites. This influenced the providers to invest more in CDN infrastructure development, since CDNs provide desired level of protection to Web sites against flash crowds.

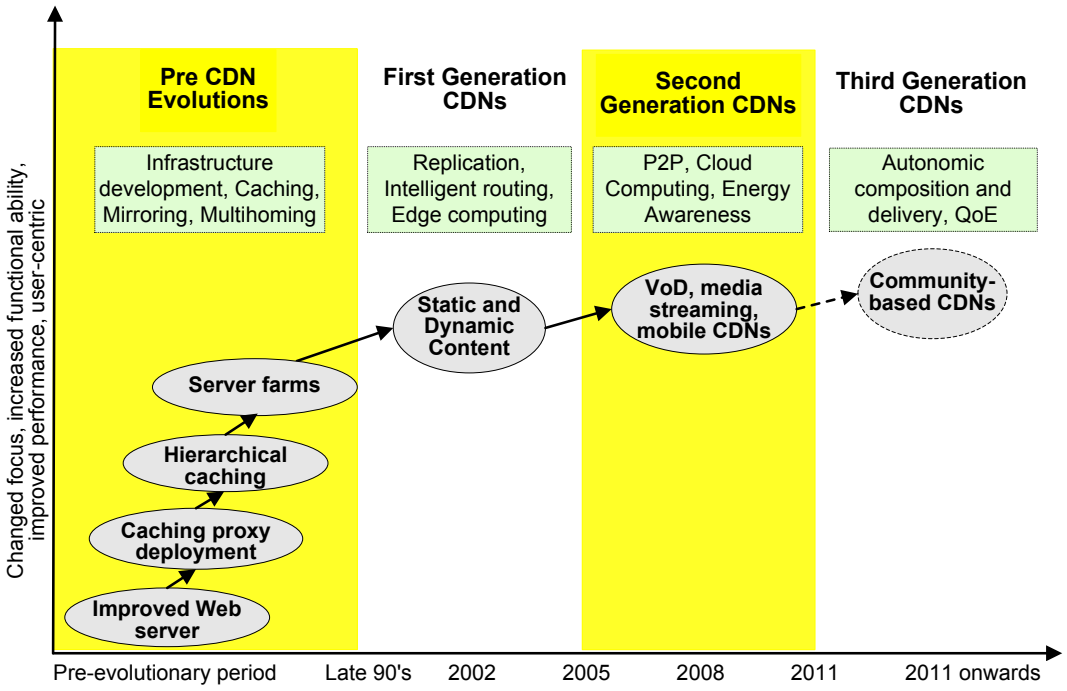


Figure 2.3: Evolution of CDN technologies.

First generation CDNs mainly focused on static and dynamic content delivery [122, 211]. The main technological feature for this generation is the development of techniques for intelligent routing, replication, and edge computing (splitting application and data across the origin and the edge). For the second generation CDNs, the focus has shifted to Video-on-Demand (VoD), news on-demand, audio and video streaming with high user interactivity. This generation has also witnessed the growth of CDNs (e.g. Ortiva Wireless) that are dedicated for content delivery to mobile de-

vices. This generation can be technologically attributed for the use of Peer-to-Peer (P2P), Cloud Computing, and energy-aware techniques [175] for content delivery and management. Although a few commercial CDNs provide capabilities of the second generation CDNs, most of the efforts on this type of CDNs are still in their infancy and have not yet fully reached the market. It is anticipated that the third generation CDNs will be community-based CDNs, i.e. they will be mainly driven by the common "people" or the average end-users. The main technological features for this generation of CDNs will be autonomic (self-configuring, self-organizing, self-managing, and self-adapting) composition and content delivery, with the main focus on meeting Quality of Experience (QoE) of end-users. There are a few research initiatives from the academic domain to develop the architecture and models in this context. More information on such community-based CDNs, along with the description of architectural components, technology, and characteristics have been described by Plagemann et al. [170]. Figure 2.3 shows the generations of CDNs and their technological features with a prediction of future evolution.

With the booming of the CDN business, several standardization activities also emerged. The Internet Engineering Task Force (IETF) as an official body has taken several initiatives through releasing Request For Comments (RFCs) [18, 20, 67, 72] in relation to the CDN research initiatives. Other than IETF, several other organizations such as Broadband Services Forum (BSF), Internet Content Adaptation Protocol (ICAP) forum, and Internet Streaming Media Alliance have developed standards for delivering broadband content and streaming of rich media content—video, audio, and associated data—over the Internet. In the same breath, by 2002, large-scale ISPs started building their own CDN functionality, providing customized services.

2.2.2 A Comparative Analysis

Data grids [140, 213], distributed databases [47, 147], and P2P networks [12, 16, 145] are three distributed systems that have some characteristics in common with CDNs. However, they differ in terms of architecture, goals, soperations, and functional characteristics. Table 2.1 provides a comparative analysis between CDNs and the related systems. Detailed description on them can be found in a previous work [164].

Table 2.1: Comparison between CDNs and related systems.

Features	CDNs	Data Grids	Distributed Databases	P2P Networks
Category	A collection of networked computers spanning the Internet	A data intensive computing environment in the form of a Virtual Organization (VO)	Locally organized collection of data distributed across multiple physical locations	Information retrieval network formed by ad-hoc resource aggregation
Constitution	Distribution of edge servers across Internet	Formation of a VO of participating institutions	Federation or splitting of existing database(s)	Collaboration among peers
Main goal	Reducing Web latency during content delivery	Performance gain through optimal and high speed data distribution from the source	Transparent integration of existing databases and replication of database fragments	File sharing among peers
Integrity	Integrity between caches	Integrity between data grid replicas	Integrity between multiple DBs	N/A
Consistency	Strong cache consistency between replicated content	Weak consistency between data grid replicas	Strong consistency between distributed databases	Weak consistency between cached content
Autonomy	None	Autonomous participants	Autonomous DDB sites	Autonomous peers
Operational activities	Content caching	Data analysis, collaboration, and maintenance across organizational and regional boundaries	Query processing, optimization, and management	Locating or caching content, encrypting, retrieving, decrypting, and verifying content
Administration	Individual companies. Proprietary in nature	Institutions who cooperate on some shared goals	Single authoritative entity	Self-interested end-users/peers

2.3 Existing CDNs: State of the Art

Current content delivery landscape consists of three types of CDNs—*commercial*, *academic*, and *cloud-based* CDNs. The CDN industry is dominated by commercial CDNs. Academic CDNs emerged as the outcome of research initiatives at various institutions. Cloud-based CDNs have recently been emerged, mainly from the commercial domain. In this section, we provide a snapshot of the current CDN domain by listing representative CDNs from each category. An updated listing can be found in the research directories of Davison [71] and Pathan [154].

Table 2.2: Summary of representative commercial CDNs.

CDN Name	Service Type	Coverage	Products/Solutions (If any)
Akamai www.akamai.com	Content delivery services, including streaming	Covers 85% of the market. 48,000 servers in 1000 networks in 70 countries. It handles 20% of total Internet traffic today	Edge Platform for handling static as well as dynamic content, Edge Control for managing applications, and Network Operations Control Center
EdgeStream www.edgestream.com	Video streaming services over the public Internet	Provides video streaming over consumer cable or ADSL modem connections worldwide	EdgeStream video on-demand and IPTV streaming software for video streaming
Limelight Networks www.limelightnetworks.com	Distributed on-demand and live delivery of video, music, games and download	Edge servers located in 72 locations around the world	Limelight ContentEdge for HTTP content delivery, MediaEdge Streaming for distributed video and music delivery, and Custom CDN for customized delivery solutions
Mirror Image www.mirror-image.com	Content delivery, streaming media, Web computing and traffic reporting	Edge servers located in 22 countries	Global Content Caching, Extensible Rules Engine, VoD, and Live Webcasting

2.3.1 Commercial CDNs

Almost all of the operational CDNs are developed by commercial companies which are subject to consolidation over time due to acquisition and/or mergers. Hence, we focus on studying only those CDNs that have been in stable operation for a significant period of time. Table 2.2 shows a list of four commercial CDNs and presents a brief summary of each of them. Detailed description of selected commercial CDNs can be found in a previous study [164].

2.3.2 Academic CDNs

Unlike commercial CDNs, the use of P2P technologies is mostly common in academic CDNs. One notable exception is the COMODIN system [83, 85] that uses a protocol to let a group of users to coordinate and control the playback of a shared multimedia

streaming session. Content delivery in an academic CDN follows a decentralized approach and request load is spread across all the participating hosts, and the system can handle node failures and sudden load surges. Academic CDNs built using P2P techniques are effective for static content only, and therefore, are unable to handle dynamically generated content due to the uncachable nature of dynamic content. Table 2.3 provides a brief summary of the representative academic CDNs. Details on academic CDNs are available in existing literature [83, 85, 87, 164, 168, 223, 228].

Table 2.3: Summary of existing academic CDNs.

CDN Name	Description	Service Type	Implementation and Testing	Availability
CoDeeN www.codeen.cs.princeton.edu	CoDeeN is an academic testbed CDN built on PlanetLab	Provides caching of content and redirection of HTTP requests	C/C++ implementation. Tested on Linux (2.4/2.6) and Mac (10.2/10.3)	N/A
COMODIN http://lisdip.deis.unical.it/research/index.html	COMODIN is a streaming CDN deployed on an international testbed	Provides collaborative media playback service	Java-based applications and applets. Tested on Unix/Linux and Windows	N/A
Coral www.coralcdn.org	Coral is a free P2P CDN. It is hosted on PlanetLab	Provides content replication in proportion to the content's popularity	C++ implementation. Tested on Linux, openBSD, FreeBSD, and Mac OS X	No official release yet. Coral is a Free software, licensed under GPLv2
Globule www.globule.org	Globule is an open source collaborative CDN. A Third party module for Apache Web server	Provides replication of content, monitoring of servers and redirecting client requests to available replicas	Implemented using PHP scripting, C/C++ and tested on Unix/Linux and Windows	Subject to a BSD-style license and the Apache software license for the packaged Apache HTTP Web server

2.3.3 Cloud-based CDNs

With the introduction of Cloud Computing[1] [15, 35], the hottest concept in IT today, the CDN landscape has experienced the growth of cloud-based CDNs or Content De-

[1] A recent technology trend that moves computing and data away from desktop and portable PCs into computational resources such as large Data Centers ("Computing") and make them accessible as scalable, on-demand services over a network (the "Cloud").

livery Clouds [60]. They extend traditional CDNs model by exploiting clouds for content and Web application delivery, as well as storage, raw computing, or access to any number of specialized services. By doing so, they endeavor to improve cost efficiency, accelerate innovations, attain faster time-to-market, and achieve application scalability [124]. There are a number of major players in this domain that are providing cloud-based content delivery services on a commercial basis, either by themselves or by partnering with an existing CDN. An example research initiative in this context is MetaCDN [29, 157], an integrated overlay network that leverages resources from existing storage clouds to provide content delivery sevices (described in Chapter 7). The main goals of the MetaCDN system is to provide economics of scale and high content delivery performance through its simple yet general purpose, reusable, and reliable geographically distributed framework.

A comparative analysis of the representative cloud-based CDNs is provided in Table 2.4. Details on them are available in the research directory of Pathan [154].

Table 2.4: Feature comparison of cloud-based CDNs.

Feature	**Amazon (S3 and CloudFront)**	**Voxel (VoxCAST and Silverlining)**	**Akamai (Cloud Optimizer)**
Storage and content delivery	S3 Storage services; CloudFront content delivery	Silverlining cloud services; VoxCAST CDN	NetStorage services; EdgePlatform content delivery
Service type	On-demand storage in multiple datacenters; on-demand content delivery	Managed hosting; On-demand content delivery	On-demand storage and content delivery
Performance	Comparable latency with customer-owned data centers. Sparsely reported performance problem due to outages	Reported consistent performance on par with competitors such as Akamai and Limelight	Up to 400% improvement and at least twice faster application response time than Amazon EC2
Availability and reliability	Availability zones to enable resiliency for single location failure, and redundancy	All time availability as it fails safe against origin server outages	No single point of failure, automatic failover and redundancy
Geographic distribution	Datacenters at 14 locations in Asia, Europe and North America	Point of Presence (POP) at 17 locations in Asia, Europe and North America	48000 servers in 1000 networks world-wide
Multi-tenancy/ Service sharing	Yes	Yes (also dedicated mode)	Yes

Feature	Amazon (S3 and CloudFront)	Voxel (VoxCAST and Silverlining)	Akamai (Cloud Optimizer)
Load balancing	Listed in future investments	Yes (server switching)	Yes (global and dynamic load balancing)
On-demand scalability	Yes	Yes	Yes
Developer API	Yes (Amazon Web services-AWS)	Yes (Hosting API)	Yes (EdgeScape API)
Automatic replication	S3: No; CloudFront: Yes	Yes	Yes
SLA (%)	99-99.9	100	100
Accessibility	AWS API or management console	VoxCAST Web-based portal	Akamai EdgeControl
Economic model and pricing	Pay-as-you-go	Progressive universal scale billing upon usage	Volume-based pricing; pay-per-use model for NetStorage
Security	Protection for DDoS attacks, access control list and firewalls	Secure authentication, firewalls	Protection for DDoS attacks and application firewall

2.4 A Taxonomy of CDNs

CDNs have received considerable research attention in the recent past [33, 98, 176, 215]. A few studies have investigated CDNs to categorize and analyze them, and explore the uniqueness, weaknesses, opportunities, and future directions in this field. Peng [166] presents an overview of CDNs. He describes the critical issues involved in designing and implementing an effective CDN, and surveys the approaches proposed in literature to address these problems. Vakali and Pallis [211] present a survey of CDN architecture and popular CDN service providers. The survey is focused on understanding the CDN framework and its usefulness. They identify the characteristics and current practices in the content networking domain, and present an evolutionary pathway for CDNs, in order to exploit the current content networking trends. Dilley et al. [76] provide an insight into the overall system architecture of the leading CDN, Akamai. They provide an overview of the existing content delivery approaches and describe Akamai's network infrastructure and its operations in detail. They also point out the technical challenges that are to be faced while constructing a global CDN like Akamai. Saroiu et al. [188] examine content delivery from the point of view of four content delivery systems: Hypertext Transfer Protocol (HTTP) Web

traffic, the Akamai CDN, Gnutella [1, 199], and KaZaa [123, 136] peer-to-peer file sharing systems. They also present significant implications for large organizations, service providers, network infrastructure providers, and general content delivery providers. Kung and Wu [118] describe a taxonomy for content networks and introduce a new class of content networks that perform "semantic aggregation and content-sensitive placement" of content. They classify content networks based on the attributes in two dimensions: content aggregation and content placement. Sivasubramanian et al. [198] identify the issues for building a Web replica hosting system. Since caching infrastructure is a major building block of a CDN and content delivery is initiated from the origin server, they consider CDNs as replica hosting systems. In this context, they propose an architectural framework, review related research work, and categorize them. A survey of peer-to-peer (P2P) content distribution technologies [12] studies current P2P systems and categorize them by identifying their non-functional properties such as security, anonymity, fairness, increased scalability, and performance, as well as resource management, and organization capabilities. Through this study the authors make useful insights for the influence of the system design on these properties. Cardellini et al. [41] study the state of the art of Web system architectures that consist of multiple server nodes distributed on a local area. They provide a taxonomy of these architectures, and analyze routing mechanisms and dispatching algorithms for them. They also present future research directions in this context.

2.4.1 Motivations and Scope

As mentioned above, there exist a wide range of work covering different aspects of CDNs such as content distribution, replication, caching, and Web server placement. However, none of them performs a complete categorization of CDNs by considering the functional and non-functional aspects. Functional aspects include technology usage and operations of a CDN, whereas the non-functional aspects focus on CDN characteristics such as organization, management, and performance issues. We set out to consider both functional and non-functional aspects of CDNs, which assist in examining the way in which the characteristics of a CDN are reflected in and affected

by the architectural design decision followed by a given CDN. We develop a comprehensive taxonomy of CDNs that identifies and categorizes numerous solutions and techniques related to various design dynamics [160].

The taxonomy is built around the core issues for building a CDN system. We identify the following key issues that pose challenges in the development of a CDN:

- ***What is required for a harmonious CDN composition.*** It includes decisions based on different CDN organization, node interactions, relationships, and content/service types.
- ***How to perform effective content distribution and management.*** It includes the right choice of content selection, surrogate placement, content outsourcing, and cache organization methodologies.
- ***How to route client requests to an appropriate CDN node.*** It refers to the usage of dynamic, scalable, and efficient routing techniques.
- ***How to measure a CDN's performance.*** It refers to the ability to predict, monitor, and ensure the end-to-end performance of a CDN.

A full-fledged CDN system design seeks to address additional issues to make the system robust, fault tolerant, secure, and capable of wide-area application hosting. In this context, the issues to be addressed are:

- ***How to handle wide-area failures in a CDN.*** It involves the use of proper tools and systems for failure detection.
- ***How to ensure security in a wide-area CDN system.*** It refers to the solutions to counter distributed security threats.
- ***How to achieve wide-area application hosting.*** It seeks to develop proper techniques to enable CDNs to perform application hosting.

Each of the above issues is an independent research area itself and many solutions and techniques can be found in literature and in practice. While realizing proper solution for the additional issues is obvious for a CDN development, our taxonomy [160] concentrates on the first four core issues. Nevertheless, Section 2.6 presents a coverage of the relevant solutions in the context of the additional issues.

2.4.2 Taxonomy

Table 2.5 summarizes a taxonomy of CDNs based on four key issues—*CDN composition*, *content distribution and management*, *request-routing*, and *performance measurement*. The first issue covers several aspects of CDNs related to their organization and formation. It classifies the CDNs with respect to their structural attributes. The next issue pertains to the content distribution mechanisms in CDNs. It describes the content distribution and management approaches of CDNs in terms of surrogate placement, content selection and delivery, content outsourcing, and caches/replica organization. Request-routing techniques in the existing CDNs are described under the next issue. Finally, the last issue deals with the performance measurement methodologies of CDNs. Detailed description, case studies, and relevant examples of the CDN taxonomy are reported in a prior publication [160].

Table 2.5: CDN taxonomy.

Issues	**Taxonomy**
CDN composition	• CDN organization (*overlay and network approach*) • Servers (*origin and replica server*) • Relationships (*client-to-surrogate-to-origin, network element-to-caching proxy, inter-proxy*) • Interaction protocols (*network elements and inter-cache interaction*) • Content/service types (*static, dynamic, streaming content and services*)
Content distribution and management	• Content selection and delivery (*full and/or partial site—empirical, popularity, object, and cluster-based*) • Surrogate placement (*single or multi-ISP approach using—center, greedy, hotspot, topology-informed, tree-based and scalable replica placement*) • Content outsourcing (*cooperative push-based, non-cooperative pull-based, or cooperative pull-based*) • Cache organization (*caching techniques—Intra- and inter-cluster caching; and cache update – periodic, on-demand, propagation, invalidation*)
Request-routing	• Request-routing algorithms (*adaptive or non-adaptive*) • Request-routing mechanisms (*DNS-based—NS and CNAME redirection, application-level and IP anycasting; transport-layer—usually combined with DNS-based techniques; and application-layer—HTTP redirection, URL rewriting, and CDN peering using centralized directory, Distributed Hash Table (DHT), or document routing model*)
Performance measurement	• Internal and external measurement (*network statistics acquisition – network probing, traffic monitoring, surrogate feedback to measure performance metrics such as geographical proximity, latency, packet loss, average bandwidth, downtime, startup time, frame rate, server load; and simulation-based performance measurement*)

The analysis of the structural attributes of a CDN reveals that its infrastructural components are closely related to each other. The structure of a CDN varies depending on the content/services it provides to its users. Within a CDN structure, a set of surrogates is used to build the content-delivery component, some combinations of relationships and mechanisms are used for redirecting end-user requests to a surrogate and interaction protocols are used for communications between CDN elements.

Content distribution and management is strategically vital in a CDN for efficient content delivery and overall performance. Content distribution includes content selection and delivery based on the type and frequency of specific end-user requests; placement of surrogates to some strategic positions so that the edge servers are close to end-users; and content outsourcing to decide which outsourcing methodology to follow. Content management is largely dependent on the techniques for cache organization (i.e. caching techniques, cache maintenance, and cache update).

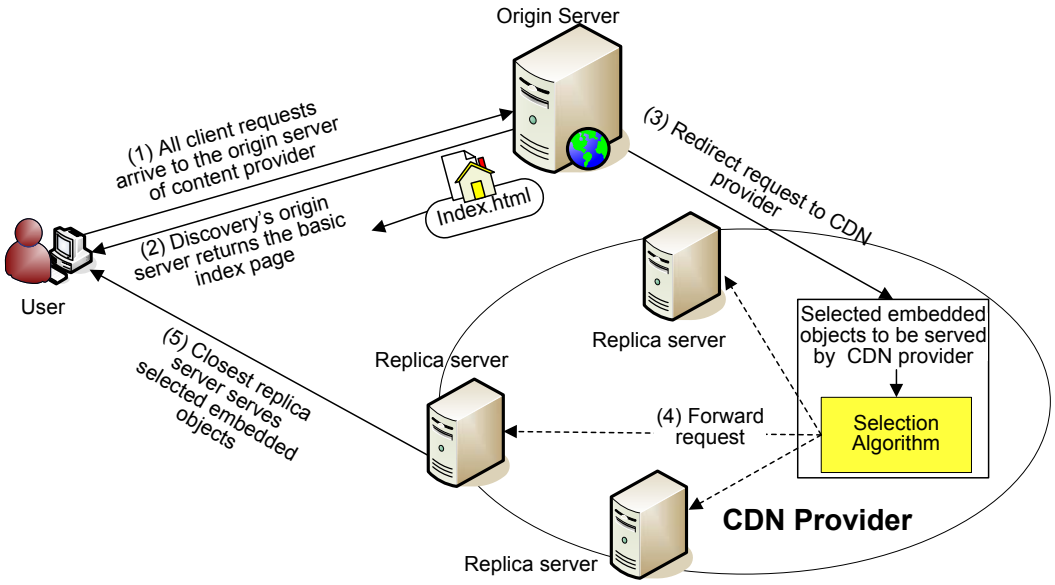


Figure 2.4: Request-routing in a CDN.

Request-routing is an indispensible enabling cornerstone for CDNs. It is generally used to direct end-user requests to replica servers based on various policies and a possible set of metrics, such as network proximity, user perceived latency, bandwidth, content availability, and replica server load. The request-routing system in a CDN deploys a request-routing algorithm and uses a request-routing mechanism

[198]. A request-routing algorithm specifies how to select an edge server in response to a given request. It is at first invoked upon receiving an end-user request. Then the user is informed about the selection using a request-routing mechanism.

Figure 2.4 provides a high-level view of the request-routing technique in a typical commercial CDN environment. The interaction flows are: (1) the end-user requests content from the content provider by specifying its URL in the Web browser. User's request is directed to the origin server; (2) when the origin server receives a request, it makes a decision to provide only the basic content (e.g. index page of the Web site); (3) to serve the high bandwidth demanding and frequently asked content (e.g. embedded objects—dynamic content, images, logos, navigation bar, and banner advertisements), content provider's origin server redirects user's request to the CDN provider; (4) using the proprietary selection algorithm, the CDN provider selects a replica server which is the 'closest' to the end-user in order to serve the requested embedded objects; (5) selected replica server gets the embedded objects from the origin server, serves the user requests, and caches it for subsequent request servicing.

Performance measurement of a CDN is done to evaluate its ability to serve the customers with the desired content and/or services. Typically five key metrics are used by the content providers to evaluate the performance of a CDN [77, 88, 104, 116, 125]. They are: cache hit ratio, reserved bandwidth, latency, surrogate server utilization, and reliability. In addition to performance measurements on CDN testbeds, researchers often use simulation tools [28, 52, 108, 205, 222] to measure a CDN's performance or experiment specific policies on overlay platforms such as PlanetLab [57].

2.5 Mapping the Taxonomy to CDNs

This section maps the above taxonomy to a few representative CDNs (Section 2.3) to demonstrate its applicability to categorize and analyze the present-day CDNs. In addition, it presents perceived insights and a critical evaluation of existing systems.

2.5.1 CDN Composition

Table 2.6 shows the annotation of representative CDNs based on the composition taxonomy. The majority of existing CDNs use overlay approach (application-specific

servers and caches for content delivery) for CDN organization, while some use network approach (network components equipped with logic for application-specific content delivery) or both. The use of both overlay and network approaches is common among commercial CDNs such as Akamai and Mirror Image. When a CDN provider uses a combination of these two for CDN formation, a network element can be used to redirect HTTP requests to a nearby application-specific surrogate server.

Most of the academic CDNs are built using P2P techniques following an overlay approach. COMODIN is an exception which uses network approach to deliver collaborative media support over IP-based networks. Each of the P2P-based academic CDNs differs in the way the overlay is built and deployed. For example, CoDeeN overlay consists of deployed "open" proxies, whereas Coral overlay (consisting of cooperative HTTP proxies and a network of DNS servers) is built relying on an underlying indexing infrastructure, and Globule overlay is composed of the user nodes.

Recently emerged cloud-based CDNs exploit an overlay approach with the use of multiple datacenters, i.e. distributed cloud infrastructure, for storage and content delivery. They differ in the way they use cloud services for content delivery. For instance, Amazon CloudFront operates with the assistance of its companion Amazon Simple Storage Service (S3) for durable storage of original, definitive version of the content, VoxCAST operates with its Silverlining cloud computing platform, and Akamai's Cloud Optimizer is built on its EdgeComputing platform consisting of a highly distributed network of edge servers.

In an overlay approach, client-to-surrogate-to-origin server and network element-to-caching proxy relationships are the most common. Inter-proxy relationship is also observed that supports cache interaction. When network approach is used, CDNs deploy request-routing logic to the network elements based on predefined policies and provide content delivery services. The overlay approach is preferred over the network approach because of the scope for new services integration and simplified management of underlying infrastructure. Offering a new service in overlay approach is as simple as distributing new code to CDN servers [122].

CDNs use origin and replica servers to perform content delivery. Most of the rep-

lica servers are used as Web servers for serving Web content. Some CDNs such as Akamai, EdgeStream, Limelight Networks, and Mirror Image use their replicas as media servers for delivering streaming media and video hosting services. Replica servers can also be used for providing caching, large file transfer, reporting, and DNS services. Alike commercial CDNs, cloud-based CDNs support origin and replica servers either through their own infrastructure (e.g. Cloud Optimizer, VoxCAST) or a companion storage cloud platform (Amazon CloudFront). P2P-based CDNs configure replicas to serve different purposes. For example, each CoDeeN node is capable of acting as a forward, reverse, and redirection proxy; Coral proxies are cooperative; and a Globule node can play the role of an origin, replica, backup, and/or replica.

Table 2.6: CDN composition taxonomy mapping.

CDN Name	CDN Organization	Servers	Relationships	Interaction Protocols	Content/Service Types
Akamai	Network and overlay approach	Origin and replica servers	Client-to-surrogate-to-origin, Network element-to-caching proxy, Inter-proxy	Network elements interaction, inter-cache interaction	Static and dynamic content, streaming media, and services (network monitoring, geographic targeting)
Edge Stream	Network approach	N/A	N/A	Network elements interaction	Video streaming and video hosting services
Limelight Networks	Overlay approach	Origin and replica servers	Client-to-surrogate-to-origin, Network-to-caching proxy	Network elements interaction	Static content, streaming media
Mirror-Image	Network and Overlay approach	Origin and replica servers	Client-to-surrogate-to-origin, Network element-to-caching proxy	Network elements interaction	Static content, streaming media, Web computing and traffic reporting services
CoDeeN	Overlay approach with "open" caching proxies	Origin and replica (forward, reverse, redirector) servers	Client-to-surrogate-to-origin, Network element-to-caching proxy, Inter-proxy	Network elements interaction, inter-cache interaction	*Participating* users receive better performance to *most* sites; only provides static content
COMODIN	Network approach	Origin and replica servers	Client-to-surrogate-to-origin	Network elements interaction	Streaming media

CDN Name	CDN Organization	Servers	Relationships	Interaction Protocols	Content/Service Types
Coral	Overlay approach with an underlying indexing infrastructure	Origin and replica (cooperative) proxy caching servers	Client-to-surrogate-to-origin, Network-to-caching proxy, Inter-proxy	Network elements interaction, inter-cache interaction	Most *users* receive better performance to *participating* sites; only provides static content
Globule	Overlay approach with end-user nodes	Origin, replica, backup and/or redirector servers	Client-to-surrogate-to-origin, Network-to-caching proxy, Inter-node	Network elements interaction, inter-cache interaction	Improved Web site performance and availability; static content delivery and monitoring
CloudFront (Amazon)	Overlay approach with multiple datacenters	Replica servers (origin servers through S3)	Network element-to-datacenter servers, Inter-proxy	Network elements interaction, inter-cache interaction	Static content and streaming media
Cloud Optimizer (Akamai)	Overlay approach with edge servers	Origin and replica (NetStorage services)	Client-to-surrogate-to-origin, Network-to-caching proxy, Inter-proxy	Network elements interaction, inter-cache interaction	Static, dynamic cloud applications and uncachable content
VoxCAST (Voxel)	Overlay approach with multiple datacenters	Origin and replica (cloud hosting servers)	Client-to-surrogate-to-origin, Network-to-caching proxy, Inter-proxy	Network elements interaction, inter-cache interaction	Static content, and live audio/video streaming

From Table 2.6, it can also be seen that most of the CDNs are dedicated to provide particular content, since variation of services and content requires the CDNs to adopt application-specific characteristics, architectures, and technologies. Most of them provide static content, while only a few of them provide streaming media, broadcasting, and other services. Cloud-based CDNs mostly provide static content delivery and limited streaming services. Most of the cloud-based CDNs (except Akamai Cloud Optimizer) are yet to realize the functionalities for dynamic and Web application delivery services. While the main business goal of commercial CDNs is to gain profit through content and/or service delivery, the goal of academic CDNs differs from each other. As for instance, CoDeeN provides static content with the goal of providing *participating* users better performance to *most* Web sites; COMODIN focuses to provide collaborative playback control services on streaming media, rather

than content delivery; Coral aims to provide most *users* better performance to *participating* Web sites; and Globule targets to improve a Web site's performance, availability, and resistance (to some extent) to flash crowds and the Slashdot effects.

Table 2.7: Content distribution and management taxonomy mapping.

CDN Name	Content Selection and Delivery	Surrogate Placement	Content Outsourcing	Cache Organization
Akamai	Content selection • Full and partial-site delivery Content Clustering • Users' sessions based	Multi-ISP approach; Hotspot placement by allocating more servers to sites experiencing high traffic load	Non-cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Propagation • On-demand
Edge Stream	Content selection • Partial-site delivery Content Clustering N/A	Single-ISP approach	Non-cooperative pull-based	Caching technique • Inter-cluster caching Cache update N/A
Limelight Networks	Content selection • Partial-site delivery Content Clustering N/A	Multi-ISP approach	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand
Mirror Image	Content selection • Partial-site delivery Content Clustering • URL based	Multi-ISP approach; Center placement following a concentrated "Superstore" architecture	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand
CoDeeN	Content selection • Partial-site delivery Content Clustering N/A	Multi-ISP approach; Topology-informed replica placement	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • On-demand
COMODIN	Content selection • Full-site delivery Content Clustering N/A	Single-ISP approach	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • Periodic update
Coral	Content selection • Full and partial-site delivery Content Clustering • Users' sessions based	Multi-ISP approach; Tree-based replica placement	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Cache invalidation

CDN Name	Content Selection and Delivery	Surrogate Placement	Content Outsourcing	Cache Organization
Globule	Content selection • Full and partial-site delivery Content Clustering N/A	Single-ISP approach; Replica placement strategy is dynamically selected by evaluating different strategies	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Adaptive cache update
CloudFront (Amazon)	Content selection • Partial-site delivery Content Clustering N/A	Multi-ISP approach with 14 datacenter locations	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Propagation
Cloud Optimizer (Akamai)	Content selection • Full and partial-site delivery Content Clustering • Users' sessions based	Multi-ISP approach with 48000 servers at distributed edge locations	Non-cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Propagation • On-demand
VoxCAST (Voxel)	Content selection • Full and partial-site delivery Content Clustering N/A	Multi-ISP approach with 17 POP locations	Non-cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Propagation

2.5.2 Content Distribution and Management

The mapping of the content distribution and management taxonomy to the representative CDNs is shown in Table 2.7. It is observed that most of the representative CDNs with extensive geographical coverage follow multi-ISP approach to place numerous number of surrogate servers at many global ISP POPs. Commercial CDNs such as Akamai, Limelight Networks, Mirror Image, and academic CDNs such as Coral [87] and CoDeeN [222] use multi-ISP approach. Cloud-based CDNs also follow this approach as they build their cloud infrastructure by placing multiple datacenters at edge locations distributed around the globe. Single-ISP approach suffers from the distant placement of surrogates with respect to the locality of end-users. However, the setup cost, administrative overhead, and complexity associated with the system deployment and management in multi-ISP approach is higher. An exception is found for high traffic volume sites. Multi-ISP approach performs better as single-ISP approach is suitable only for low-to-medium traffic volume sites [211].

Content outsourcing of the commercial CDNs mostly use non-cooperative pull-based approach because of its simplicity enabled by the use of DNS redirection or URL rewriting. Cooperative push-based approach is still theoretical and none of the existing CDNs supports it. Cooperative pull-based approach involves complex technologies, e.g. Distributed Hash Table (DHT), as compared to the non-cooperative approach and it is used by the P2P-based academic CDNs [152]. Moreover, it imposes a large communication overhead (in terms of number of messages exchanged) when the number of clients is large. It also does not offer high fidelity when the content changes rapidly or when the coherency requirements are stringent.

From Table 2.7 it is also evident that representative commercial, academic and cloud-based CDNs with large geographic coverage use inter-cluster (and a combination of inter and intra-cluster) caching. CDNs mainly use on-demand update as their cache update policy. Only Coral uses invalidation for updating caches since it delivers static content which changes very infrequently. Globule follows an adaptive cache update policy to dynamically choose between different cache consistency enforcement techniques. Of all the cache update policies, periodic update has the greatest reach since the caches are updated in a regular fashion. Thus, it has the potential to be most effective in ensuring cache content consistency. Update propagation and invalidation are not generally applicable as steady-state control mechanisms, and they can cause control traffic to consume bandwidth and processor resources that could otherwise be used for serving content [92]. Content providers themselves may administer to deploy specific caching mechanisms or heuristics for cache update. Distributing particular caching mechanism is simpler to administer but it has limited effects. On the other hand, cache heuristics are a good CDN feature for content providers who do not want to develop own caching mechanisms. However, heuristics do not deliver the same results as well-planned policy controls [92].

Table 2.8: Request-routing taxonomy mapping.

CDN Name	Request-routing Technique
Akamai	• Adaptive request-routing algorithms which takes into account server load and various network metrics • DNS-based and application-layer (URL rewriting) request-routing

CDN Name	Request-routing Technique
EdgeStream	Application-layer (HTTP redirection)
Limelight Networks	DNS-based request-routing
Mirror-Image	Global Server Load Balancing (GSLB) • Global awareness • Smart authoritative DNS
CoDeeN	• Transport-layer (request-routing algorithm takes into account request locality, system load, reliability, and proximity information) • Application-layer (HTTP redirection)
COMODIN	DNS-based request-routing
Coral	• Transport-layer (request-routing algorithms to consider locality by exploiting on-the-fly measurement and storing topology hints) • DNS-based request-routing
Globule	• Adaptive request-routing algorithms considering AS proximity • Single-tier DNS-based request-routing
CloudFront (Amazon)	• DNS-based request-routing (CNAME redirection) • Application-layer (HTTP redirection)
Cloud Optimizer (Akamai)	• DNS-based request-routing • Application-layer request-routing (proprietary)
VoxCAST (Voxel)	DNS-based request-routing

2.5.3 Request-Routing

Table 2.8 maps the request-routing taxonomy to the representative CDNs. It can be observed from the table that DNS-based mechanisms are very popular for request-routing. The main reason of this popularity is its simplicity and the ubiquity of DNS as a directory service. DNS-based mechanisms mainly use a specialized DNS server in the name resolution process. Among other request-routing mechanisms, HTTP redirection is also highly used in the CDNs because of the finer level of granularity on the cost of introducing an explicit binding between a client and a replica server. Flexibility and simplicity are other reasons to use HTTP redirection for request-routing in CDNs. Some CDNs such as Mirror Image uses GSLB for request-routing. It is advantageous since less effort is required to add GSLB capability to the network without adding any additional network devices. Transport-layer request-routing is mostly used in academic CDNs. Among them, Coral exploits overlay routing techniques, where indexing abstraction for request-routing is done using Distributed Sloppy Hash Table (DSHT). Thus, it makes use of P2P mechanism

for request-routing. The request-routing algorithms used by the commercial CDNs are proprietary in nature. The technology details of most of them have not been revealed. Our analysis of the existing CDNs indicates that Akamai and Globule use adaptive request-routing algorithms for their request-routing system. Akamai's adaptive (to flash crowds) request-routing takes into account server load and various network metrics; whereas Globule measures only the number of Autonomous Systems (AS) that a request needs to pass through. In CoDeeN, the request-routing algorithm takes into account request locality, system load, reliability, and proximity information. Coral's request-routing algorithm improves locality by exploiting on-the-fly network measurement and storing topology hints in order to increase the possibility for the clients to discover nearby DNS servers.

Table 2.9: Performance measurement taxonomy mapping.

CDN Name	**Performance Measurement**
Akamai	Internal measurement • Network probing • Traffic monitoring (proactive) External measurement • Performed by a third party (Giga Information group)
EdgeStream	Internal measurement • Traffic monitoring through Real Time Performance Monitoring Service (RPMS)
Limelight Networks	Internal measurement • Traffic monitoring (proactive) and reporting External measurement • Support for Omniture measurement system
Mirror-Image	Internal measurement • Network probing • Traffic monitoring and reporting
CoDeeN	Internal measurement • Local traffic and system monitoring
COMODIN	Internal measurement • Network probing and feedback from surrogates
Coral	Internal measurement • Traffic monitoring • Liveness checking of a proxy via UDP RPC
Globule	Internal measurement • Traffic monitoring • Monitoring of server availability by the redirectors

CDN Name	Performance Measurement
CloudFront (Amazon)	Internal measurement • Traffic monitoring, reporting and usage logging External measurement • Performed by end-users [154]
Cloud Optimizer (Akamai)	Internal measurement • Network probing • Traffic monitoring (proactive) Performance also measured externally
VoxCAST (Voxel)	Internal measurement • Traffic statistics, usage reporting and historical logs

2.5.4 Performance Measurement

Table 2.9 shows the mapping of different performance measurement techniques to representative CDNs. Performance measurement of a CDN through some metric estimation measures its ability to serve the customers with the desired content and/or services. The performance of a CDN should be evaluated in terms of cache hit ratio, bandwidth consumption, latency, surrogate server utilization, and reliability. In addition, other factors such as storage, communication overhead, and scalability can also be taken into account. The estimation of performance metrics gives an indication of system conditions and helps for efficient request-routing and load balancing in large CDN systems. It is important to a content provider to analyze the performance study results of a CDN so as to select the most appropriate CDN provider for itself. However, the proprietary nature of CDN providers usually does not allow a content provider to conduct performance measurement on them.

From Table 2.9, we see that performance measurement of a CDN is done through internal measurement technologies as well as from the customer and end-user perspective. It is evident that most of the CDNs use internal measurement based on network probing, traffic monitoring, or the like. Akamai uses proactive traffic monitoring and network probing for measuring performance. In the academic domain, CoDeeN has the local monitoring ability that examines a service's primary resources, such as free file descriptors/sockets, CPU cycles, and DNS resolver service; COMODIN controls the collaborative streaming performance via network probing and surrogate feedback; Coral performs a proxy's liveness check (via UDP Remote Pro-

cedure Call (RPC)) prior to replying to a DNS query; whereas, Globule has implemented monitoring ability in its redirector servers to check server availability.

External performance measurement of CDNs is not common because most of the operating CDNs are commercial enterprises, which are not run transparently, and there are corporate advantages to keep the performance metrics and methodologies undisclosed. Despite this, some CDNs such as Akamai and Limelight Networks allow third-party external measurements.

2.6 Discussion

As stated in Section 2.4.1, a full-fledged CDN development requires addressing additional issues (other than the four core issues considered for the taxonomy) such as fault tolerance, security, and the ability for Web application hosting. In this section, we present a brief discussion on them and assist CDN researchers to comprehend respective fields by providing referral to relevant research materials.

Being CDNs a complex fabric of distributed network elements, failures can occur at many places. Following a concentrated architecture such as local clustering may improve fault-tolerance. However, it creates a single-point of failure, when the ISP connectivity to the cluster is lost. This problem can be solved through deploying Web clusters in distributed locations (mirroring) or using multiple ISPs to provide connectivity (multihoming). While clustering, mirroring, or multihoming address the CDN robustness issue to some extent, they introduce additional problems. Clustering suffers from scalability, while mirroring requires each mirror to carry entire load, and multihoming requires each connection to carry the entire traffic. Commercial CDNs follow their own proprietary approaches to provide fault-tolerance and scalability. As for instance, Akamai has developed a distributed monitoring service to ensure that server or network failures are handled immediately without affecting end-users. There are numerous existing solutions, some of which are used in real systems. Interested readers are referred to existing literature [102, 172, 189] to find descriptions on the explicit fault-tolerance solutions in wide-area systems such as CDNs.

Ensuring security in CDNs pose extra challenges in system development. There

are security concerns at different levels of a CDN such as network, routers, DNS or Web clusters. One common security threat is the DDoS attack. The DDoS attack can be aimed at (a) consumption of scarce resources such as network bandwidth or CPU; (b) destruction or modification of configuration information; and (c) physical destruction or modifications of network components [184]. Security threats also include attacks that exploit software vulnerabilities (intrusion attacks) and protocol inconsistencies (protocol attacks). There exist many research addressing wide-area security problems. Extensive coverage and documentation of related solutions are available in the literature [91, 105-107, 111, 114, 184].

Nowadays, commercial CDNs such as Akamai provide usage-based content and application delivery solutions to end-users. Akamai Edge Computing Infrastructure (ECI) [76], Active Cache [40], and ACDN [177] replicate the application code to the edge servers without replicating the application data itself. Rather, the data is kept in a centralized server. This approach enables the Web tier applications to extend to the CDN platform so that end-user requests for application object would be executed at the replica server rather than at the origin. However, this approach suffers from increased wide-area latency due to excessive data access and bottleneck due to the concentration on a centralized server. To overcome these limitations, Sivasubramanian et al. [196] propose an approach for replicating Web applications on-demand. This approach employs partial replication to replicate data units only to the servers who access them often. In another work, application-specific edge service architecture [90] is presented where the application itself is responsible for its replication with the compromise of a weaker consistency model. More information on hosting wide-area applications can be found in an analysis of Web applications caching and replication strategies by Sivasubramanian et al. [197].

2.7 Positioning the Thesis

Along with providing an operational understanding for a CDN, the proposed taxonomy and the covered technologies establish the background for this thesis. Our work on CDN peering (Chapter 1) is targeted to provide larger scale (aggregate in-

frastructure size), reach (diversity of content locations), and sustainability of a CDN provider than they could otherwise achieve. However, there are limits to how large any CDN's scale and reach can be. Essentially a given CDN's scale or reach is limited by its operational cost, geographical coverage, and the demand for that scale/reach of its infrastructure [72]. Often these factors put constraints on a CDN's existence.

Our investigation on the current CDNs reveals that the CDN industry is getting consolidated as a result of acquisitions and/or mergers. Moreover, with the introduction of recent IT trends such as Cloud Computing, content delivery, caching, and replication techniques are gaining more attention in order to meet up the new technical and infrastructure requirements for the next generation CDNs. CDN peering is an answer to deal with such consolidations and new technology integrations, with improved coverage and performance for CDN providers. Identifying new issues, innovations, and challenges in the design, architecture and development of such an advanced content delivery service requires better understanding and interpretation of the essential concepts in this area. Therefore, the in-depth analysis of CDN technologies and comprehensive comparison framework based on the presented taxonomy will not only serve as a tool to understand this complex area, but also will help to map future research efforts in CDNs.

The issues considered in the taxonomy provide us a guideline for developing CDN peering solutions. We follow an overlay approach to develop an architecture (Chapter 3) for CDN peering by exploiting available and underutilized resources from multiple CDNs. The proposed architecture encloses the structural attributes of a CDN—servers, relationships, and interaction protocols. The developed architecture is mainly focused on static content delivery. However, it can be extended to deliver dynamic, streaming media content, and services.

In the thesis, we have limited focus on the issues for content distribution and management. However, at times we have made a few assumptions to implicitly take them into account in order to ensure efficient content delivery and improved performance. Our work is applicable for both full and partial site content selection and delivery. Depending on the demand for scale and reach, we may follow a single or

multi-ISP approach for placing surrogates. We further seek to use a cooperative push-based approach to outsource content to the servers of the participating CDNs. We have conducted performance modeling (Chapter 4) and discrete-event simulations (Chapter 5), based on Independent Replication Method [42, 64, 121], with the assumption that each participating CDN holds a copy of the content to be served in its contributing servers. We follow a hierarchical approach for load index dissemination where server state information is reported to the CDN gateway using an asynchronous feedback mechanism. While the use of any specific technique for cache organization is out of scope for this thesis, inter-cluster caching with periodic or on-demand update can be anticipated within the CDN peering architecture.

We present an adaptive request-redirection algorithm that takes traffic load and network proximity into account (Chapter 5). Within the CDN peering architecture, request assignments endeavor to make use of both DNS and gateway redirection. Thus, we address the need to handle dynamically changing conditions such as flash crowds and other unanticipated events. Furthermore, we exploit a combination of DNS and application-layer techniques to develop and implement a utiliy-based request-redirection for the MetaCDN overlay system [157] (Chapter 7).

Along with simulations, we conduct *proof-of-concept* testbed measurement to evaluate the performance of the develop models and request-redirection mechanisms. We utilize traffic monitoring and surrogate feedback to measure a number of performance metrics such as response time, incoming traffic load, network proximity, throughput, and network utility.

Dealing with wide-area network failures, security threats, usage-based content and application delivery do not fall within the scope of this thesis. Therefore, we refrain from developing solutions for them. In Section 2.6, we have provided brief discussion on these issues along with necessary references to existing work.

2.8 Summary and Conclusion

In this chapter, we have presented a state-of-the-art survey of existing CDNs. After analyzing the CDN landscape, we have categorized CDNs according to the func-

tional and non-functional attributes. We have developed a comprehensive taxonomy based on four issues: CDN composition, content distribution and management, request-routing, and performance measurement. We further develop taxonomies for each of these paradigms to classify the common trends, solutions, and techniques in content networking. Additionally, we identify three issues, namely, fault tolerance, security, and ability for Web application hosting as to introduce challenges in CDN development. Hereby, we provided pointers to related research work in this context. In doing so, we assist CDN researchers to gain insights into the technology, services, strategies, and practices that are currently followed in this field. We have also performed a mapping of the taxonomy to representative CDNs. Such a mapping provides a basis to realize an in-depth analysis and complete understanding of the CDN domain, and to validate the applicability and accuracy of the taxonomy.

In the next chapter, we present an architecture that enables coordinated and cooperative content delivery via *internetworking* between multiple content delivery services to rapidly "scale-out" to meet both flash crowds [14] and anticipated increases in demand, and remove the need for a given CDN to over-provision its resources.

Chapter 3

CDN Peering

Existing Content Delivery Networks (CDNs), operating in isolation, are often prone to Service Level Agreement (SLA) violations and resources over-provisioning in order to ensure high quality services to end-users, thus incurring extensive operational cost and labor. As mentioned in Chapter 1, CDN peering [34, 155, 156, 161] is an approach to reduce expenses and avoid adverse business impact. It is formed by a set of autonomous CDNs who cooperate through a mechanism to share resources while enjoying larger scale and reach. This chapter presents a novel architecture that establishes the basis to form CDN peering. This chapter also outlines a comparative analysis with related research and presents two complementary models to assist peering. Finally, it lists the research challenges to realize the presented architecture.

3.1 Introduction

CDN peering exhibits a negotiated "resource sharing" relationship among multiple providers. It offers necessary distributed computing and network infrastructure to handle unexpected or anticipated traffic load spikes in demand, when a single provider has insufficient resources. The initiating provider to form a CDN peering arrangement is called a *primary*, whereas other CDNs who agree to share resources are called *peers*. In the CDN peering system, a provider serves user requests as long as the incoming traffic load can be handled internally. If the load exceeds its capacity, the excess requests are offloaded to other least loaded Web server(s) of its peers, while still upholding user perceived performance. Each CDN has its own user request stream and a set of Web servers, but delegates only a subset of them, i.e.

subCDN, to take part in peering. The endpoint of a peering negotiation between two CDNs is a contract (SLA) that specifies the peer resources (servers, bandwidth etc) that will be allocated to serve content on behalf of the primary. The primary directly manages the resources it has acquired, insofar that it determines what content is served and what proportion of the incoming traffic is redirected. At any time, a given CDN may be in the roles of either a primary or a peer, i.e. the roles are fluid. The participants in peering act as a single conceptual unit in the execution of common goal(s). A content request to the CDN peering system further initiates activities that users are unaware of, e.g. inter-provider request-redirection, content replication and delivery.

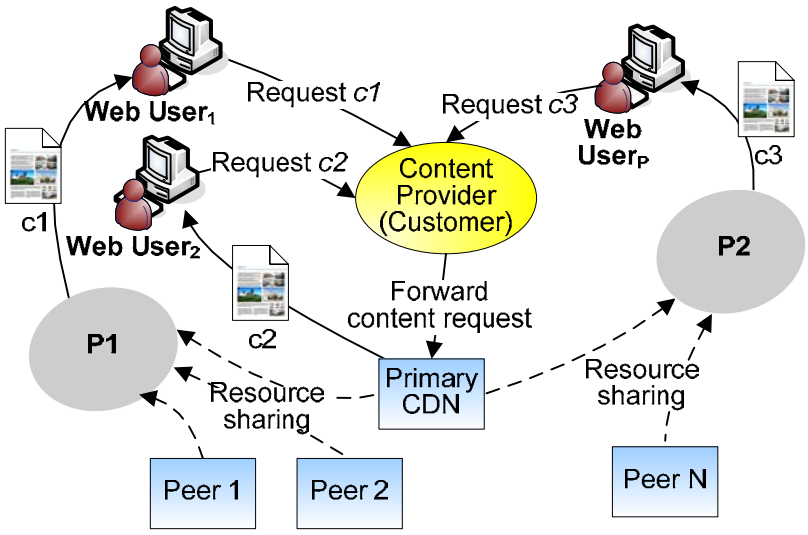


Figure 3.1: Example of CDN peering.

A CDN peering arrangement comprises *explicit* members—a primary and peers, who cooperate for resource sharing, and *implicit* members, who are content providers and end-users. While implicit members are transparent to peering, they share benefits from it. Figure 3.1 shows an example with two peering arrangements, P1 and P2. User requests are forwarded to the primary that is holding the content from the content provider. The requested content is served either directly by the primary or by a peer. Let us consider content c1 and c3 in the figure. Since c1 and c3 reside in the servers within P1 and P2 respectively, requests for c1 and c3 are served from P1 and P2. In case of content c2, the primary directly delivers the requested content.

3.2 CDN Peering Initiatives

Interconnecting multi-provider content delivery capabilities is gaining popularity in the research community, due to its flexibility and effectiveness to improve performance for end-users and to achieve pervasive geographical coverage and increased capacity for a provider. In this section, we provide a comparative analysis of research related to CDN peering in order to ascertain the feasibility of our architecture and position it as the basis to address the shortcomings of prior initiatives.

Related research efforts found in literature can be separated into different groups: architectural models vs. analytical performance models, and provider vs. end-user side perspective to assist peering. We start with investigating the architectures for enabling the peering concept and then present related performance modeling approaches. We also mention the initiatives in this context from user-side perspective along with the deployed Peer-to-Peer (P2P) CDNs.

The Content Distribution Internetworking (CDI) [72] model allows CDNs to have a means of affiliating their delivery and distribution infrastructure with other CDNs that have content to distribute. Each provider treats its neighbors as *black boxes*, and uses commonly defined internetworking protocol, while using its proprietary protocol internally. While the CDI model lays the foundation for interconnecting providers, it provides only an abstract view of how this interconnection could be formed. Based on the CDI model, Turrini [209, 210] presents a protocol architecture, where performance data is interchanged between CDNs before forwarding a request by an authoritative CDN (for a particular group). This approach adds an overhead to the user perceived response time. Moreover, being a point-to-point protocol, if one end-point is down the connection remains interrupted until that end-point is restored. Since no evaluation is provided for performance data interchange, the effectiveness of the protocol is unclear. Our work in this thesis is complementary to the CDI model in that we present a CDN peering architecture, and devise mechanisms to enable peering among providers.

Several research efforts follow the CDI initiative. Specifically, a brokerage system, called CDN Brokering [24], is developed and deployed on the Internet on a provi-

sional basis. It is based on an Intelligent Domain Name Server (IDNS), a specialized request-routing DNS server, with a proprietary routing mechanism that allows one CDN to intelligently redirect end-users to other CDNs in that domain. The aim of this work is not to devise mechanisms for load sharing among providers or to explicitly consider end-user perceived performance to satisfy QoS while serving requests, but to demonstrate the usefulness of brokering. We differ by following a peering approach with the focus on improving end-user experience via dynamic load distribution using request-redirection. Our practical work in this context (Chapter 7) is rather in line with the brokering concept.

Cardellini et al. [45] presents an architecture for geographically distributed Web systems that integrates DNS proximity and dispatcher scheduling with an HTTP redirection mechanism. It aims to minimize the response time experienced by users while accessing geographically distributed Web sites. One potential drawback of this work is the use of HTTP request redirection, which may lead to increased network impact on user perceived response time. Our approach is significantly different as they do not capture the heterogeneity in Web server systems and only consider homogeneous Web clusters belonging to a single entity.

While the above mentioned research do not explicitly virtualize multiple providers, Amini et al. [10] present a peering system for content delivery workloads in a federated, multi-provider infrastructure. The core component of the system is a peering algorithm that directs end-user requests to partner providers to minimize cost and improve performance. While this work is appealing, the peering strategy, resource provisioning, and QoS guarantees between partnering providers are unexplored in this work.

There also exist several research to develop analytical models for geographically distributed Web servers. Among them an approach to model traffic redirection in geographically diverse server sets [9] and Wide Area Redirection of Dynamic Content (WARD) [182] can be mentioned. The first uses a novel metric, called Server Set Distance (SSD) to simplify the modeling and classification of redirection schemes. The latter presents an architecture for redirecting dynamic content requests from an

overloaded Internet Data Center (IDC) to a remote replica. Both of them do not consider multiple providers in the developed models.

From a user-side perspective, Cooperative Networking (CoopNet) [148] provides cooperation of end-hosts to improve network performance perceived by all. This cooperation between end-users is invoked for the duration of a flash crowd. CoopNet is found to be effective for small Web sites with limited resources, but sacrifices transparency to end-users. Therefore, it can not be used as a replacement or alternative for cooperation among infrastructure-based CDNs.

Among the deployed P2P-based CDN systems, CoDeeN [151, 223], CoralCDN [87], and Globule [167, 168] can be mentioned. CoDeeN provides content delivery services, driven entirely by end-user demands. However, utilizing its services is not transparent to end-users, as they require them to "opt-in" by setting their browser proxy manually to interact with the CoDeeN network. This user-driven approach means that CoDeeN is essentially an enhanced caching mechanism rather than a true CDN. While CoDeeN may cooperate with external content providers, it is yet to be explored.

CoralCDN utilizes a P2P DNS approach to direct users to replica nodes in the CoralCDN overlay network, reducing the stress on origin servers and improving performance for end-users. It is a cooperative network, but there is no means for nodes (or providers) to participate in peering with nodes that are outside of PlanetLab. The nodes that can participate are only offered a coarse level control over their participation (such as allowing individual servers to specify their maximum peak and steady-state bandwidth usage) but there is no fine grained control over exactly what content a node has agreed to serve, nor are there service guarantees.

Globule is a third-party module for Apache Web server to provide services of a collaborative CDN. It enables server-to-server peering, ad-hoc selection, creation, and destruction of replicas, consistency management and relatively transparent redirection (via HTTP or DNS) of end-users to high-performing replicas. A brokerage service is offered where participants can register and access other participants' details in order to initiate negotiations. While such negotiations could be aided with

necessary muti-provider peering mechanisms, this has not been explored deeply in Globule.

DotSlash [229] is a community driven "mutual" aid service to offer support for small sites that would not have the resources to cope during flash crowds. Provided the site in question has configured itself to access DotSlash, the service automatically intervenes during the flash crowd, allocating and releasing "rescue" servers depending on the load, and is phased out once the flash crowd passes. We draw similarity with the above mentioned collaborative CDNs; however, we show uniqueness by capturing the notion of multi-provider existence for content delivery.

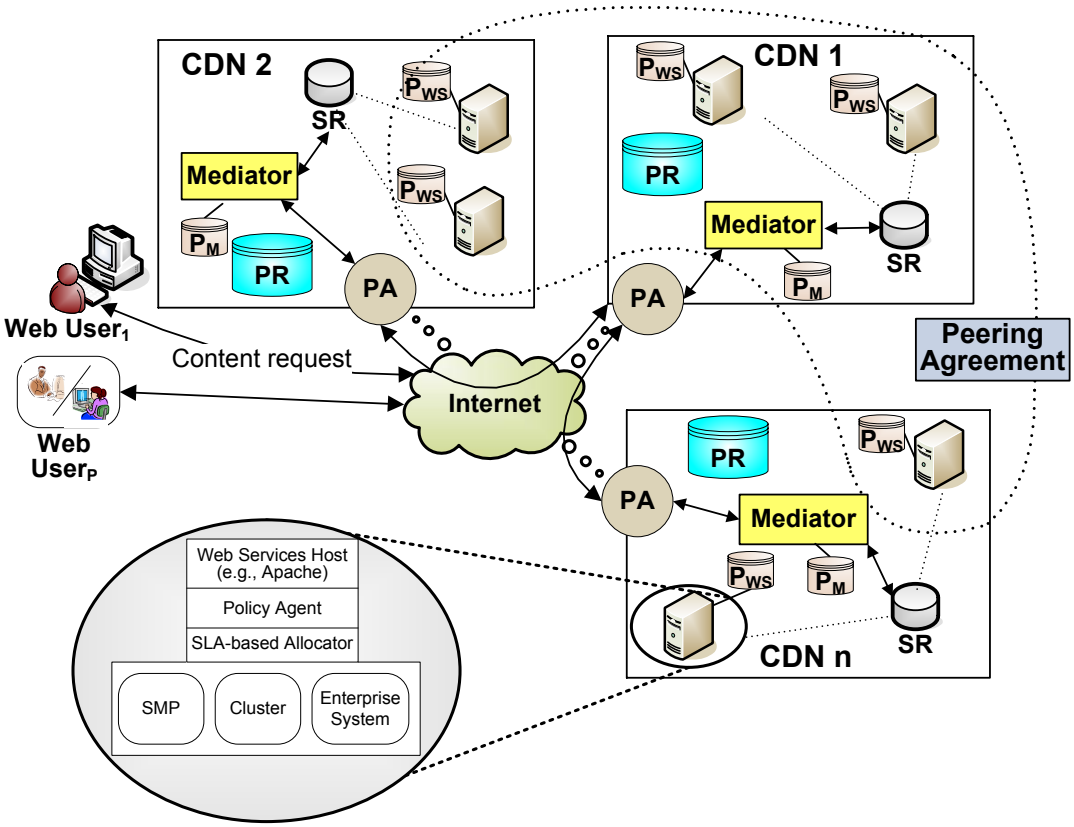


Figure 3.2: Architecture of a system to assist the creation of CDN peering.

3.3 System Architecture

Figure 3.2 presents the cooperative architecture of a system for CDN peering. The terminologies used to describe the system are listed in Table 3.1.

Table 3.1: List of commonly used terms.

Terminology	Description
Web server (WS)	A container of content
Mediator	An authoritative entity for policy negotiation and management
Service registry (SR)	Discovers and stores local resource and policy information
Peering Agent (PA)	A resource discovery module in the CDN peering environment
Policy repository (PR)	A storage of Web server, mediator, and peering policies
P_{WS}	A set of Web server-specific rules for storage and management
P_{M}	A set of mediator-specific rules for interaction and negotiation
$P_{Peering}$	A set of rules for creation and growth of the peering arrangement

3.3.1 Architectural Components

In the following we describe the core components of the proposed architecture along with their responsibilities:

- ***Web server.*** Web servers are responsible for storing content and delivering them reliably. We separate a server's structure into two layers: *overlay* and *core*. In the overlay layer, a server comprises a Web-service host (for example, Apache or Tomcat), a policy agent, and an SLA-allocator. The Web-service host ensures the delivery of content to end-users based on the negotiated policies. The policy agent is responsible (in conjunction with the mediator) for determining which resources can be delegated and under what conditions (policies) delegation is permitted. The SLA-allocator performs the provisioning and reservation of Web server's resources (e.g. CPU, bandwidth, storage etc.) to satisfy both local and delegated SLAs, and ensures that the terms of the SLAs are enforced. The Web server's core consists of high performance computing systems such as symmetric multiprocessors, cluster systems, or other enterprise systems (such as desktop grids). The Web server's underlying algorithms perform on-demand caching, content selection, and routing between servers. This requires each Web server to express its own policies (P_{WS}) for storage and management of content.
- ***Mediator.*** The (resource) mediator is a policy-driven authoritative entity in a peering arrangement. The raison d'être for an instance of the mediator is to

ensure that the participating entities are able to adapt to changing circumstances (agility) and are able to achieve their objectives in a dynamic and uncertain environment (resilience). Once a peering arrangement is formed, the mediator, on behalf of the primary, controls what portion of the Web traffic (i.e. end-user requests) is redirected to the Web servers of the CDN peering system, which content is replicated there, how the replication decision is taken, and which replication policies are being used. When performing automated negotiation the mediator directs any decision made during peering negotiations, policy management, and scheduling. A mediator holds the initial policies (P_M) for peering formation and obtains additional composite polices ($P_{Peering}$) as a result of successful negotiations. A mediator works in conjunction with its local Peering Agent (PA) to discover external resources and to negotiate with other CDNs. An example of a mediator led negotiation is given in Section 3.3.3.

- ***Service Registry (SR).*** The SR encapsulates the resource and service information for each CDN. It helps in discovering local resources through enabling the Web servers of CDN providers to register and publish their resource, service and policy details. In the face of traffic surges, the SR is accessed by the mediator to supply any necessary local resource information. When a new peering arrangement is formed, an instance of the SR is created that encapsulates all local and delegated external CDN resources' information.
- ***Policy Repository (PR).*** The PR virtualizes all the policies within a peering arrangement. It includes the Web server-specific policies, mediator policies, peering formation policies along with any policies for resources delegated as the result of peering. These policies are a set of rules to administer, manage, and control access to shared resources. They provide a way to manage the components in the face of complex technologies.
- ***Peering Agent (PA).*** The PA is a CDN specific entity that exists prior to the formation of CDN peering. It is independent of any peering arrangement. It acts as a policy-driven resource discovery module and is used as a conduit or gateway (with mediator and PR as a single conceptual entity) to exchange pol-

icy and resource information with other CDNs. It is used by the primary CDN to discover its peers' (external) resources, as well as to let them know about the local policies and service requirements prior to the commencement of actual negotiation by the mediator.

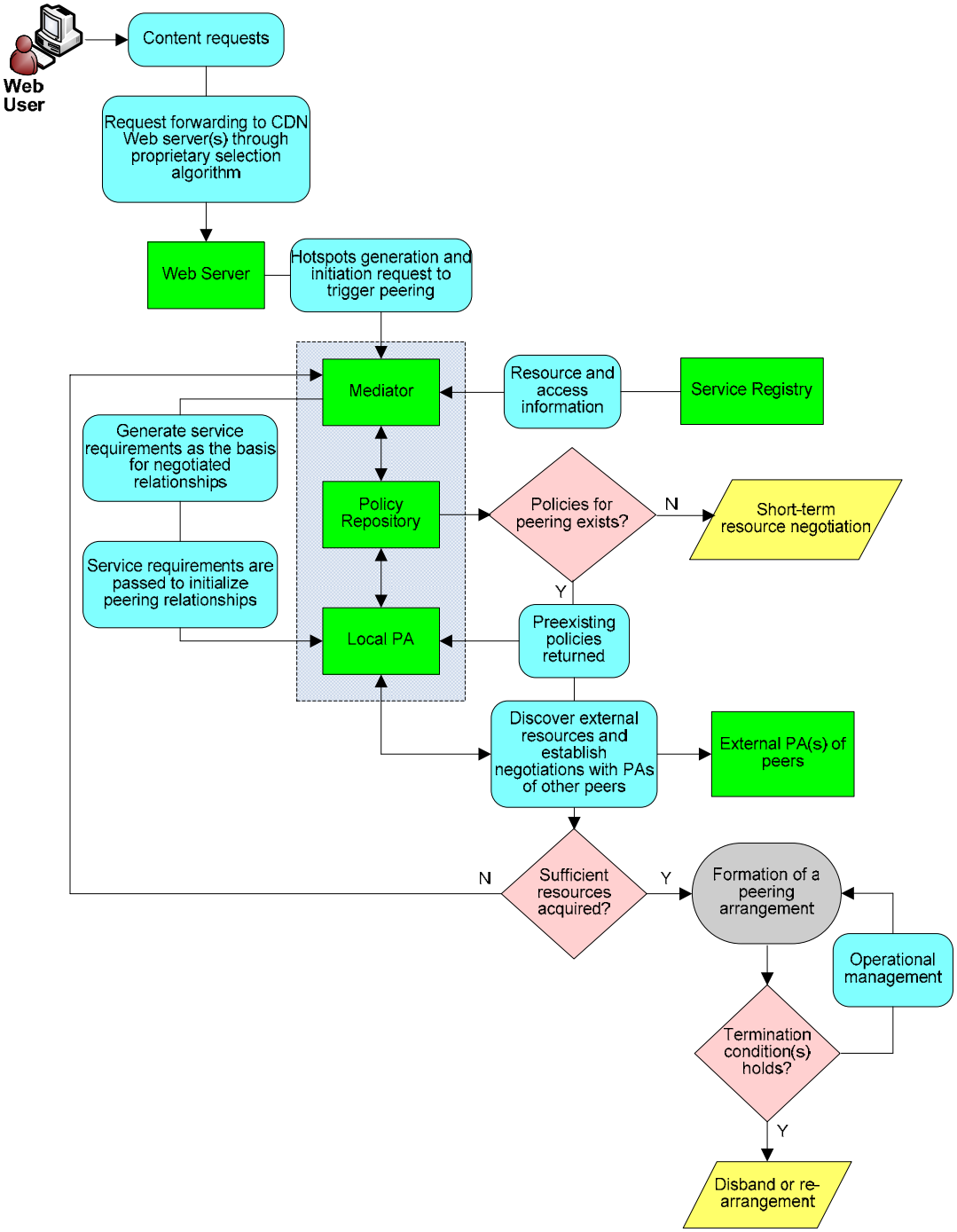


Figure 3.3: Interaction flows in a CDN peering arrangement.

3.3.2 Peering Lifecycle

CDN peering can be *short-term* wherein CDNs operate to handle flash crowds, or *long-term* in which they explore the delivery of specialized services. A short-term CDN peering is formed on-demand (detailed description is available in Section 3.3.3) and it phases out when the workload returns to normal. On the other hand, a long-term CDN peering is formed for an event which is well known in advance.

Figure 3.3 shows the interaction flows to form a CDN peering arrangement. A peering lifecycle goes through six major steps, as described in the following:

1. ***Initiation.*** Peering is triggered by an initialization request sent to the mediator under exceptional circumstances, e.g. flash crowds, when the (primary) CDN realizes that it cannot handle a part of the workload on its Web Servers. The triggering condition must consider the expected and unexpected load increases in the primary CDN.

2. ***Negotiated relationship.*** The controlling interest of a CDN to interconnect with other CDNs leads to the creation of a negotiated relationship. This relationship describes the expected level of services from the involving parties such as storage requirements, the required rate of transfer, cost of services; the expected duration of receiving service, penalties for service violations, and other preconditions such as the primary CDN's preference to gain resources at a particular region. Negotiated relationships can be established by means of non-technical terms (financial statement) or technical terms (SLAs). Thus, negotiated relationships must specify the interactions among entities including service administration, coordination and disband (or re-arrangement) of peered CDNs. In our architecture, the mediator instance obtains the resource and access information from the SR, whilst SLAs and other policies from the PR. The mediator instance on the primary CDN's behalf generates its service requirements based on the current circumstance and SLA requirements of its customer(s). Therefore, divergent policies are allowed that specify the information that can be shared during interaction through providing a certain level of visibility to preserve privacy.

3. ***Resource discovery.*** Once the primary CDN identifies its roles and activities through the negotiated relationships for coordination and cooperation between CDNs, the next step is to choose potential CDNs to peer with. The mediator instance passes the service requirements to the local PA to discover external resources from peers. The PA performs the resource discovery process through predicting performance of peers, working around issues of separate administration and limited information sharing among enlisted CDNs. If there are any preexisting peering policies (for a long term scenario) then these will be returned at this point. Otherwise, a short-term negotiation (Section 3.3.3) is carried out with PA identified peering targets.

4. ***CDN peering protocols.*** The next step is to configure the 'CDN peering' protocols at the conduit or gateway of the respective CDNs in order to technically support the terms and policies implicitly specified through the negotiated relationships. This step includes advertising the configurations (topology aspects, geographical proximity, capability, and performance) of the enlisted CDNs through inter-PA communication. Upon the establishment of a peering arrangement, these protocols also allow participating CDNs to exchange information regarding the content availability and assist to redirect requests to an optimal peer. Request-redirection in a peering arrangement depends on the content distribution and request-redirection policies (specified in the CDN peering protocols) associated with the content as well as the specific algorithms and methods used for directing these requests.

5. ***Operational management.*** When the primary CDN acquires sufficient resources from its peers to meet the SLAs with the customer, the new peering arrangement becomes operational. Once a peering arrangement is established, all participating parties cooperate in the execution of common goal(s). Peering enables providers to exchange accounting information to perform billing based on the negotiated relationships. If no CDN is interested to continue peering, re-negotiation is resumed by tuning the negotiated relationships with reconsidered service requirements.

6. ***Disband or re-arrangement.*** An existing peering arrangement may need to either disband or re-arrange itself (within the scope of the negotiated relationships) if any of the following conditions hold: (a) the circumstances under which the arrangement was formed no longer hold; (b) peering is no longer beneficial for the participating CDNs; (c) an existing peering arrangement needs to be expanded further in order to deal with additional load; or (d) participating CDNs are not meeting their agreed upon contributions.

3.3.3 Short-term Resource Negotiation

In order to respond to hotspots that may result in a CDN failing to meet its QoS obligations, we propose that time-critical agreements for a short-term peering should be automatically negotiated. We expect any such agreements to hold for a limited duration and only involve a restricted set of CDN resources. Even so, there are crucial issues involving such agreements including trust, i.e. who governs the allocation decisions and are they trustworthy; and the potential commercial sensitivity of information about the current state and costs of a CDN. Divulging commercially sensitive information (e.g. resources, access and policies) as the basis for negotiating a peering arrangement would be, in general, commercially unacceptable. Even with the limitations placed on resources that can be automatically delegated; there must be checks and balances to ensure that any delegation is made properly. Otherwise, it would also be unlikely that a CDN would agree to have an external party (e.g. mediator of the primary CDN) make allocations of their resources and bind them to negotiated SLAs.

One solution to these problems is to utilize a cryptographically secure auction [31], which hides both the valuations that CDNs place on their resources and who is participating in the auction. With this approach, no CDN needs to trust a particular auctioneer; rather the auction mechanism itself is trustworthy due to the cryptographic guarantees. Thus, malicious behavior by any CDN (including the primary) or a third party auctioneer is prevented and sensitive bid prices are kept secret. Moreover, resource over-provisioning by harnessing data through fake membership, or

modifying and falsifying of content by some rogue CDN providers is not allowed. These auctions have been shown to be tractable in practice and are therefore an ideal basis for the automation of peering agreements [31].

A negotiation would start with the formation of CDN peering, upon determining the shortfall in resources in the primary CDN. The mediator then issues its local PA a call for bids and within this includes the SLAs that it requires. The PA distributes the request to other PAs of cooperating peers and each of them then passes the request to its mediator. All mediators that wish to bid then register with the requesting mediator and a subset of the mediators (acting as the *distributed auctioneers*) are selected via a cut and shuffle [144]. Note that the requesting mediator does not act as an auctioneer. The auction is thus held securely and only the final resource allocations are revealed. The auction is combinatorial as multiple resource types can be specified and therefore different groupings and even substitutions of those resources over CDN peering are possible.

An auctioneer starts an auction not for selling an item (i.e. allocation), but for buying it. Buyers (peers) bid with the price they are willing to sell the allocation of their Web servers. One bidder can not see the bid of other bidders. An auctioneer gathers bids from the bidders and selects the lowest bidding agent(s) as the winner and the winner is paid second-lowest bidding price. In other words, a reverse Vickrey auction [217] is used.

Figure 3.4 shows the flowchart for autonomic negotiation through auction. In the following we provide a summary of the auction steps [158]:

1. ***Phase I: Auction initiation.*** Auction starts when the mediator instance (M^i) of the primary CDN (buyer) initiates peering to handle flash crowds. It internally determines the maximum payable amount (expressed by *Payoff Value*). The mediator then issues its local PA a call for bids including its service requirements (*Auction Policy*). The PA distributes the bidding request to other PAs of the peers and each of them passes the request to its mediator.
2. ***Phase II: Bidding.*** All mediators that wish to bid then register with the requesting mediator. All bidders (peers) use a *Bidding Function* to determine the

bidding amount and potential peers bid their price.

3. ***Phase III: Auction termination and re-negotiation.*** An auctioneer collects bids and they are then evaluated securely by the distributed auctioneers to determine the winners and the optimal combinations of resources. A winner is paid by the amount of the second-lowest bid. At this point, it can be assumed that a CDN has acquired sufficient resources from its peers to meet its SLAs with customers. If no winner is selected, re-negotiation through auction takes place, starting from Phase I.

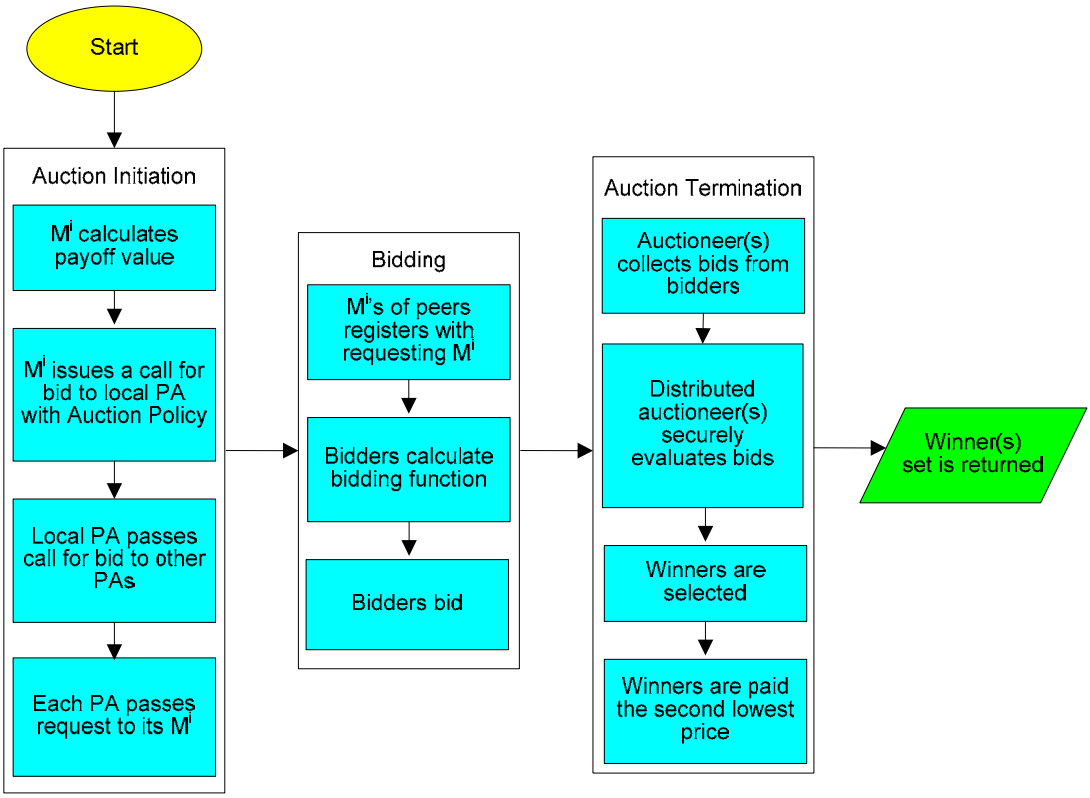


Figure 3.4: Flowchart for a short-term resource negotiation through auction.

3.3.4 SLA Negotiations

During the formation of CDN peering, participating CDNs sign SLAs with different performance objectives. Once the SLAs have been agreed upon, they work in order to satisfy the negotiated SLAs. The SLA components include:

- ***Description of service requirements.*** A specification of the resource and service requirements of the primary CDN. This description includes the storage requirements, the required transfer rate, the primary CDN's preference to gain resources at a particular region, and the expected duration of receiving service.
- ***Administration for VO activities.*** It specifies the role of the mediator as an authoritative entity in a CDN peering arrangement.
- ***Renegotiation for problem resolution.*** It illustrates the steps to be undertaken in the face of any problem in providing necessary services.
- ***Consequences of SLA violation.*** It outlines the possible results of SLA violation when service expectations are not met. The consequences of SLA violation may range from imposing penalty on the peers through reimbursement of part of the revenues lost due to the loss of service, to termination of the peering relationship, and to disbanding and/or rearranging the peering arrangement.
- ***SLA bypassing conditions.*** It details the conditions under which SLAs are not applicable. Such situations include the damage of physical resources due to natural disaster, theft etc.

3.3.5 Policy Management

To operate effectively according to SLA specifications, the CDN peering architecture seeks for the consistent availability and usage of operational policies. A policy is a descriptive logical rule that allows an entity to properly administer, manage, and execute its activities. Policies in our context are rules that specify how the Web servers should deal with different traffic types, what type of content can be delivered, what resources can be shared between peers, what measures are to be taken to ensure QoS based on negotiated SLAs, and what operations are to be performed at the origin server. These policies provide a way to manage multiple entities deploying complex technologies within the CDN peering architecture.

A policy-based framework can simplify the complexities involved in the operation and management of a CDN [216]. Hence, our architecture follows the basic policy framework of the IETF/DMTF [224] (Figure 3.5).

Table 3.2 provides a mapping of the CDN peering architecture with the basic policy framework. The PR from Figure 3.2 virtualizes the Web server, mediator and peering policies. These policies are generated by the policy management tool used by the administrator of CDN peering. The distribution network and the Web server components (i.e. Web Services host, Policy Agent, SLA-based Allocator) are the instances of the Policy Enforcement Points (PEPs), which enforce the policies stored in the repository. The mediator, as an instance of the Policy Decision Points (PDPs), specifies the set of policies to be negotiated at the time of collaborating with other CDNs, and passes them to the PA at the time of negotiation. The policy management tool is administrator dependent and it is not shown in the CDN peering architecture (Figure 3.2).

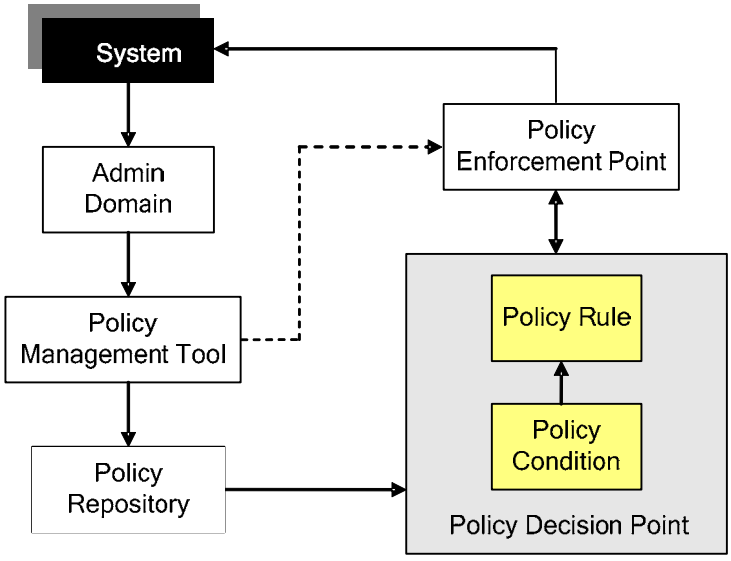


Figure 3.5: Basic policy framework.

Our proposal follows a general implementation of policies that can be specified according to their granularity level. Being influenced by Verma et al. [216], we also argue for storing the specification of the policy rules in the PR for maintaining interoperability. We initiate simplified management by storing the definition of policies in the PR, and by defining abstractions that provide a machine dependent specification of policies. We propose that the policy specification contained in the PR should be defined in terms of the technology to which a policy would apply, rather than in terms of the configuration parameters of any specific resource.

Table 3.2: Policy components mapping.

Policy component	CDN peering component	Specified policies	Description
System	CDN peering	All policies in the system	The distributed computing and network infrastructure
Admin domain	Formed peering arrangement	Negotiated VO policies	Administrative entity for resource management and access control
Policy management tool	Administrator dependent	–	An administrator dependent tool to generate policies
Policy repository	Policy repository	Server, peering and mediator policies	Storage of policies in the system
PEPs	Web server components	Web server policies	A logical entity which ensures proper enforcement of policies
PDPs	Mediator	Mediator policies, peering policies	Authoritative entity for retrieving policies from the repository

Now we delineate different levels of policies, namely—*Web server policy (P_{WS})*, *mediator policy (P_M)* and *peering policy ($P_{Peering}$)* that are present within the CDN peering architecture. Specifically, Web server policies include:

- how a server performs consistent content caching;
- how the policy agent module operates based on negotiated SLAs; and
- how the SLA-based allocator module ensures resource provisioning to satisfy negotiated SLAs.

Mediator policies specify:

- how the mediator interacts with the PA to pass service requirements;
- how the mediator takes over the administration of delegated resources once they are acquired;
- how the mediator effectively manages activities in a peering arrangement to cope with changing circumstances; and
- how the mediator coordinates with other entities for resource allocation.

Peering policies include:

- the policies necessary for the formation of a peering arrangement;
- the policies need to be administered in face of SLA violation by peers; and
- the policies to dynamically disband or rearrange a peering arrangement.

Table 3.3: Comparison of CDN models.

Features	Typical CDN models		Advanced models for CDN peering		
	Conventional CDNs	**P2P-based CDNs**	**CDN peering**	**Brokering**	**Customized brokering**
Nature of Content Delivery	Based on Web server Collaboration	Based on content availability	Based on CDN internetworking/peering	Based on CDN performance	Based on user defined QoS (Customized)
Operational responsibility	CDN Provider	Peers/ Users	Primary CDN Provider	Content Provider	Content Provider
Entities in agreement	CDN-Content Provider	Self-interested users, no agreement	CDN-Content Provider, CDN-CDN	CDN-Content Provider	CDN-Content Provider
Agreement nature	Static	N/A	Short-term or long-term	Policy-based	Dynamic
Scalability	Limited	High	High	High	High
External CDN Cooperation	No	No	Yes	Yes	Yes
Cooperation among CDNs	No	No	Yes	No, work in parallel	No, work in parallel
Cooperation between users	No	Yes	No	No	No

3.4 Alternative Models for CDN Peering

In this section we propose two additional models that can be used as alternatives to form CDN peering. They are brokering-based and QoS-driven (customized) brokering-based models. They can be used to complement the CDN peering architecture presented in Section 3.3. To better understand the uniqueness of these models and to compare them with existing ones, we provide a comparative analysis in Table 3.3.

3.4.1 Brokering-based CDN peering

Figure 3.6 shows the first of the two models that we propose to assist the creation of CDN peering. In this model, "cooperative" content delivery is achieved by the content provider (or any third party service on its behalf), who leverages the services of multiple CDNs to ensure appropriate geographical coverage and that performance targets are met. The content provider has the responsibility for efficient content delivery. The interaction flows are: (1) an end-user requests content from the content

provider by specifying its URL in the Web browser. User's request is directed to content provider's origin server; (2) the content provider utilizes a brokering system of its own in order to select CDN(s) for delivering content to end-users. A given content provider can select multiple CDNs (based on a CDN's QoS performance, capabilities, current load, and geographical location) for delivering content to its users. The selected CDNs do not need to be aware that they are working in parallel with each other, as the content provider (or a third party) handles the management and separation of responsibilities; (3) a *policy-based* agreement between the content provider and the selected CDN(s) is established; (4) once peering is established, the proprietary algorithm of the selected CDN(s) chooses optimal server to deliver desired content to the end-user.

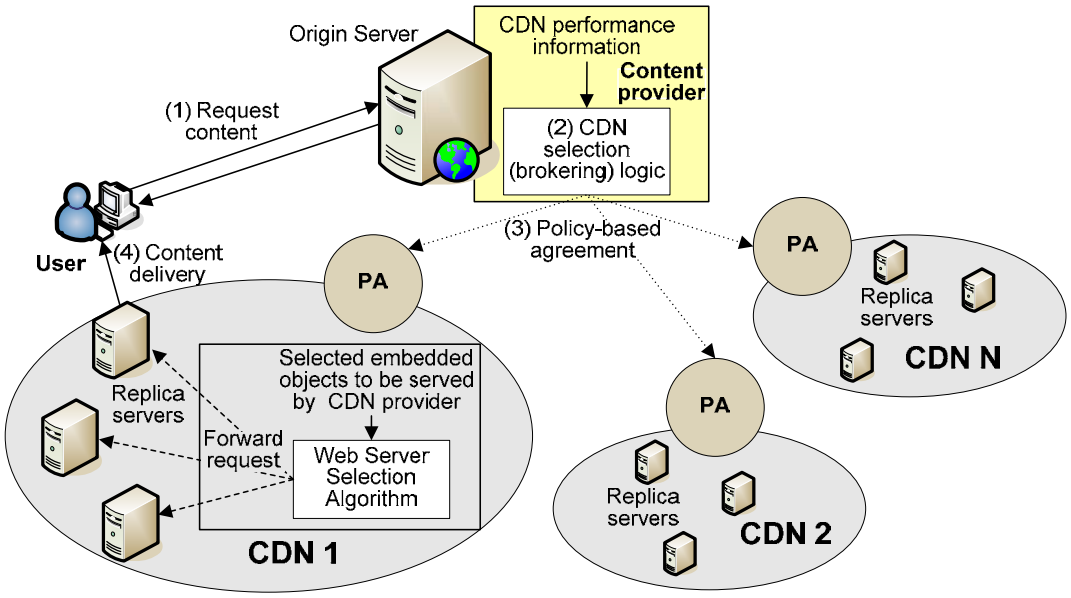


Figure 3.6: Brokering-based approach to form CDN peering.

In order to join in a peering arrangement according to this model, CDN providers have to provide improved performance. Content provider will keep track of the performance of CDNs. Hence, selection of CDN(s) can be based on history information of the performance for similar content. This model also allows a content provider to offer preferential treatment to its users based on a certain policy (can be as simple as "receive service according to payment" or any other complex policy).

This model is applicable in practical context. Specifically, it has been used to de-

velop a third-party cloud-based content delivery service, called MetaCDN [29, 157], which leverages multiple storage cloud providers' resources to ensure high performance content delivery. We elaborate on the MetaCDN system and performance measurements on its global testbed in Chapter 7.

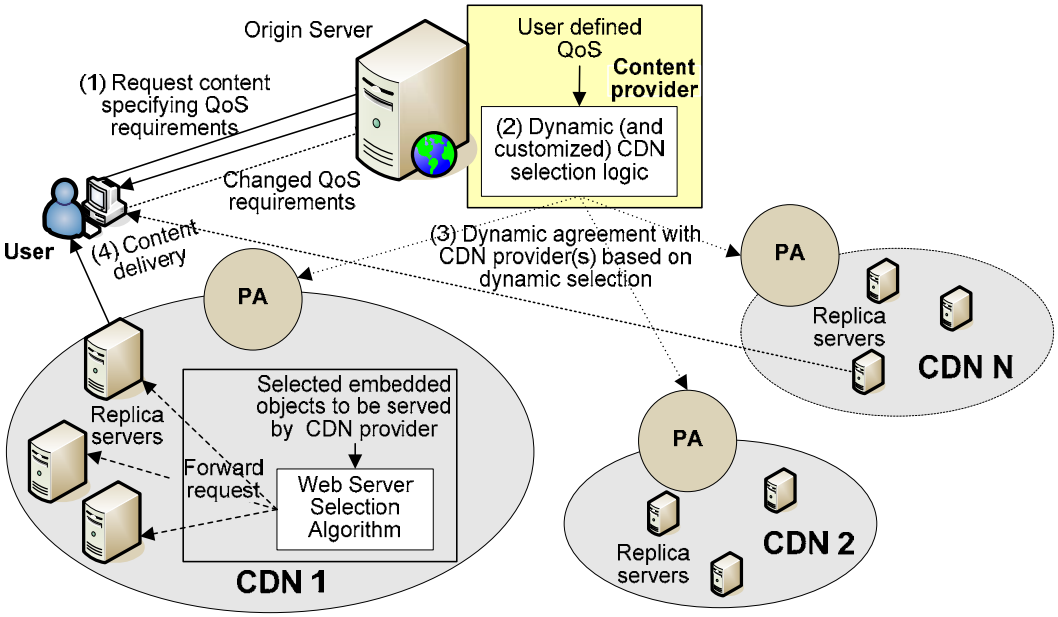


Figure 3.7: QoS-driven brokering-based approach for CDN peering.

3.4.2 QoS-Driven (customized) Brokering-based CDN peering

While the above model considers the performance of each potential participant to form CDN peering, it does not specifically consider the QoS required by end-users. Users can have dynamic requirements depending on situations (e.g. flash crowds) that will "customize" content delivery. Therefore, sophistication on user-defined QoS is required to be adopted in the model, which may depend on the class of users accessing the service. Hence, in Figure 3.7 we show an improvement on the previous model to assist CDN peering formation. In this model, the content provider performs the participant selection dynamically based on the individual user (or a group of users) QoS specifications. The interaction flows are: (1) an end-user requests content from the content provider with specific QoS requirements and it reaches the content provider's origin server; (2) content provider uses a dynamic algorithm (based on

user-defined QoS) to select CDN(s); (3) content provider establishes *dynamic* agreement with the CDNs it utilizes to ensure that end-user QoS targets are met; (4) once peering is established with the selected CDN(s), desired content is delivered from the optimal Web server of the selected peer(s).

Such peering arrangements are user-specific and they vary in terms of QoS target, scope, size, and capability. It is evident that content provider has the responsibility for effective content delivery through dynamic peering arrangements. Thus, if a particular peering arrangement fails to meet the target QoS to effectively deliver content to end-users, content provider re-negotiate with the CDN providers to establish new peering arrangement(s). In Figure 3.7, we show that in the initial peering arrangement, CDN 1 is responsible for delivering content to users. As the QoS requirements change (shown in dotted line), content provider revokes the CDN selection logic to establish a new peering arrangement. In the reformed peering arrangement, CDN N is the new participant that delivers content to the end-users from its Web server.

It should be noted that brokering-based CDN peering can also be formed at the client-side, where brokering is performed by the end-user according to his/her service requirements. It can be illustrated with a scenario where a content provider's streaming media player chooses one among many CDNs based on various performance criteria including QoS. Premium content is streamed over a premium high performing CDN, while non-premium content is served through a cheap less performing CDN (or even streamed directly from origin). Depending on the need, the end-user chooses where to stream from.

3.5 Challenges to Realize CDN Peering

There are a number of challenges, both technical and non-technical (i.e. commercial and legal), that can block the rapid growth of CDN peering. In this section, we outline some of the common stoppers for the uptake of CDN peering.

- ***Legal/copyright issues.*** There can often be complex legal issues associated with the content to be delivered (e.g. embargoed or copyrighted content) that could prevent CDNs from arbitrarily cooperating with each other. Interactions be-

tween peering partners must consider any legal issues associated with the content to be served when delegating it to participating mirror servers from different CDN providers. For instance, if a content provider needs some software or documents that contained logic or information that was embargoed by certain governments, i.e. its access is restricted, all participating CDN providers would have to ensure this was enforced to comply with the appropriate laws. Currently, academic CDNs such as CoDeeN and Coral offer little to no control on the actual content a participating node delivers, and as such participants in these systems could be inadvertently breaking these laws. Content that is copyrighted needs to be carefully managed to ensure that the copyright holder's rights are respected and enforced. The operation (e.g. caching and replication) of some CDNs are user-driven rather than initiated by the content provider, who would prefer to distribute their content on their own terms rather than have it populated in caches worldwide without their consent.

- ***Global reach.*** The most common scenario for CDN providers is a centrally managed, globally distributed infrastructure. Companies such as Akamai and Limelight Networks have their own far-reaching global networks that cover the vast majority of their customers needs. Indeed, their pervasive coverage is essentially their competitive advantage, and allows them to target the higher end of the customer market for these services. However, few providers can match their global reach, and as such they have little commercial or operational incentive to peer with other smaller providers.
- ***Consolidation in CDN market.*** Direct peering might be advantageous for small CDN providers, if they wish to compete with larger providers based on coverage and performance. In recent years there has been an enormous consolidation of the CDN marketplace from 20-30 providers down to 5-10 providers of note [154, 164, 186]. It is clear that smaller providers found it difficult to compete on coverage and performance with Akamai and Mirror Image, and subsequently ceased operation or were acquired by the larger providers. CDN peering should restrict such acquisition and/or merger.

- ***Challenges in brokering-based CDN peering.*** An approach where a content provider itself (or a third-party on its behalf) manages the selection and contribution of many CDNs to distribute its content seems appealing, especially, if they have the resources and the know-how to manage such an effort. CDN providers could be chosen on their respective merits (e.g. locality, performance, price) and their efforts combined together to provide a good experience for their customers. However, enforcing QoS to ensure a good end-user experience could be challenging when dealing with multiple providers, especially when they are not actually collaborating, rather simply operating in parallel. In this context, client-side brokering-based solution can be handy to solve end-user performance problem and QoS issues.
- ***Challenges in P2P-based CDN peering.*** There has been a growing trend in the last decade toward exploiting user-side bandwidth to cooperatively deliver content in a P2P manner. Whilst initially this started against the wishes of content providers (e.g. Napster, Gnutella), eventually content providers embraced P2P technology, in particular BitTorrent, in order to distribute large volumes of content with scalability and performance that vastly exceeded what was possible with a traditional globally distributed CDN. Content providers have utilized this effectively to distribute digital media (movies, music), operating systems (e.g. Linux) and operating systems patches, games and game patches. With end-user bandwidth increases as a result of the proliferation of high-speed broadband, content providers leverage the masses, which upload data segments to peers as they download the file themselves. However, this approach is only effective for popular files, and can lead to poor end-user experience for a content that is not being 'seeded' by enough users. As such, it is difficult for content providers to guarantee any particular QoS bounds when the nodes distributing the content are simply end-users themselves that may have little motivation to cooperate once they have received their data.
- ***Lack of incentives for cooperation.*** Further complicating the widespread dependence of a P2P-based peering approach is a backlash by Internet Service

Providers (ISPs) that are unhappy with the content providers pushing the burden and cost of content delivery onto end-users (and subsequently the ISPs themselves). Many ISPs are now actively blocking or throttling BitTorrent and other P2P traffic in response to this trend, to minimize increased utilization and reduction in revenue per user and the resulting cost it places on the ISP in provisioning additional capacity. Many ISPs in more geographically isolated countries, such as Australia and New Zealand are in particularly unique situations, depending on a small number of expensive data pipes to North America and Europe. Hence, the broadband access offered by ISPs in these regions have fixed data quotas (rather than 'unlimited') that users are restricted to, in order to ensure they remain profitable. These conditions may discourage the widespread adoption and participation by users in cooperative content delivery.

3.6 Technical Issues for CDN Peering

In this section, we present some of the unique technical issues for CDN peering that are to be addressed from a research perspective. While some solutions exist for related problems in CDNs, the notion of peering poses extra challenges. We provide a research pathway by highlighting the key research questions to realize CDN peering.

3.6.1 Load Distribution for CDN Peering

The load distribution strategy for CDN peering includes *request assignment* and *redirection*, *load dissemination*, and *content replication*. Coordination among them is another important consideration for successful exploitation of the load distribution strategy.

Request assignment and *redirection* to geographically distributed Web servers of peers requires considering an end-user's location, server loads, and link utilization between the end-user and server in addition to task size, i.e. processing requirements of a content request. It should also address the need to handle dynamically changing conditions, such as flash crowds and other unpredictable events. Request assignment and redirection can be performed in a CDN at multiple levels—at the DNS, at the gateways to local clusters and also (redirection) between servers in a cluster [46, 63]. Commercial CDNs predominantly rely on DNS level end-user assignment combined

with a rudimentary request assignment policy (such as weighted round robin, or least-loaded-first) which updates the DNS records to point to the most appropriate replica server [76]. In the CDN peering architecture, end-users can be assigned via DNS (by the PAs of participating CDNs updating their DNS records regularly) and also via redirection at the CDN gateway when appropriate.

To deal with the *load dissemination* issue, the behavior of traffic can be modeled under expected peak load since in this case the server load is most severely tested. Load information can be measured and disseminated within individual CDNs and among other CDNs. A load index can provide a measure of utilization of a single resource on a computer system. Alternatively, it can be a combined measure of multiple resources such as CPU load, memory utilization, disk paging, and active processes. Such load information needs to be disseminated among all participating CDNs in a timely and efficient manner to maximize its utility. Such indices will also be crucial to identify situations where forming a peering arrangement is appropriate (e.g. when servers or entire CDNs are overloaded) or when CDNs' resources are underutilized and could be offered to other CDN providers. Within the CDN peering architecture, a hierarchical approach can be anticipated, where current bandwidth and resource usage of CDN servers is reported to the CDN gateway in a periodic or threshold-based manner. The gateways of participating CDNs then communicate aggregated load information describing the load of their constituent servers.

Content replication occurs from origin servers to other servers within a CDN. Existing CDN providers (e.g. Akamai, Mirror Image) use a non-cooperative pull-based approach, where requests are directed (via DNS) to their closest replica server [76]. If the file requested is not held there, the replica server pulls the content from the origin server. Co-operative push-based techniques have been proposed that push content onto participating mirror servers using a greedy-global heuristic algorithm [43]. In this approach, requests are directed to the closest server, or if there is no suitable mirror nearby, they are directed to the origin server. In the context of our architecture, this replication extends to participating servers from other CDNs in a given peering arrangement, subject to the available resources it contributes to the collaboration.

In summary, the following questions are to be addressed to perform load sharing in a CDN peering arrangement:

- How to deduce a dynamic request assignment and redirection strategy that calculates ideal parameters for request-routing during runtime?
- How to ensure reduced server load, less bandwidth consumption (by particular CDN server) and improve the performance of content delivery?
- How do participating CDNs cooperate in replicating content in order to provide a satisfactory solution to all parties?
- What measures can be taken to ensure that the cached objects are not out-of-date? How to deal with uncacheable objects?

3.6.2 Coordination of CDNs

Any solution to the above core technical issues of load distribution must be coordinated among all participants in a peering arrangement in order to provide high performance and QoS. A cooperative middleware should be developed to enable the correct execution of solutions developed to address each core issue. Related to this issue, the key question to be addressed is:

- What kind of coordination mechanisms need to be in place to ensure the effectiveness, scalability and growth of CDN peering?

3.6.3 Service and Policy Management

Content management in the CDN peering architecture should be highly motivated by end-user preferences. Hence, a comprehensive model for managing the distributed content is crucial to avail end-user preferences. To address this issue, content can be personalized to meet specific user's (or a group of users) preferences. Alike Web personalization [137], user preferences can be automatically learned from content request and usage data by using data mining techniques. Data mining over CDN can exploit significant performance improvement through dealing with proper management of traffic, pricing and accounting/billing in CDNs [152]. In this context, the following questions need to be addressed:

- How to make a value-added service to an accessible infrastructure service?
- What types of SLAs are to be negotiated among the participants? What policies can be generated to support SLA negotiation?
- How can autonomous policy negotiation happen in time to form a time-critical peering arrangement?

3.6.4 Pricing of Content and Services

Sustained resource sharing between participants in a peering arrangement must ensure sufficient incentives exist for all parties. It requires the deployment of appropriate pricing, billing, and management mechanisms. The key questions are:

- What mechanisms are to be used for value expression (expression of content and service requirements and their valuation), value translation (translating requirements to content and service distribution) and value enforcement (mechanisms to select and distribute different content and services)?
- How do CDN providers achieve maximum profit in a competitive environment, yet maintaining the equilibrium of supply and demand?

3.7 Summary and Conclusion

This chapter presents an architecture to form CDN peering, which endeavors to balance a CDN's service requirements against the high cost of deploying customer dedicated and therefore over-provisioned resources. In addition, this architecture promotes extended scalability and resource sharing with other CDNs through cooperation and coordination, thus evolving past the current landscape where disparate CDNs exist. We also present two alternative models to assist peering and identify the associated research challenges.

To realize the concept of CDN peering, we develop necessary mechanisms for optimal resource discovery, server selection, and request-redirection. Specifically, in the next chapter, we start with presenting analytical models that demonstrate the effects of peering and predict user perceived performance. This model-based approach provides the foundation to perform effective peering among multi-provider CDNs.

Chapter 4

Performance Modeling

Analytical modeling of a system provides a clearer understanding of the associated research problems, thus assists in the efficient design and analysis of the system. In this chapter, we develop Quality of Service (QoS)-driven performance models based on the fundamentals of the queuing theory to demonstrate the CDN peering effects and characterize user perceived performance. In this model, an overloaded CDN re-directs a fraction of its incoming requests to its peers and avoids the impact of flash crowds. This approach assists in making concrete QoS guarantee for a CDN.

4.1 Introduction

The definition of QoS relates to the agreements between a service provider and its customers, i.e. SLAs [26], which results in a fixed set of well understood terms—in our case, the QoS requirements of end-users. Quality can be defined from the following three different perspectives and views [73]:

- ***Quality as functionality.*** According to this view, quality is considered in terms of the amount of functionality that a service provider offers to its customers. It characterizes the design of a service and can only be measured by comparing the service against other services offering similar functionalities. For example, if CDN A provides content delivery service for static content only, while CDN B provides the same for both static and dynamic content; then CDN B can be considered to offer better quality than CDN A.
- ***Quality as conformance.*** In this view, quality is seen as being synonymous with meeting providers commitments and specification such as SLAs. Quality

as conformance, which can be monitored for each service individually, usually requires the users' experience of a service in order to measure the 'promise' against the 'delivery'. For example, if CDN A makes the commitment to its customer that 95% of the end-user requests will be served within less than 2 seconds, and it maintains it at all times in its operation, then CDN A is usually considered as offering good QoS.

- ***Quality as reputation.*** In this view, quality is linked to the users' perception of a service in general. This perception is developed gradually over the time of a service provider's existence. Quality as reputation can be regarded as a reference to a service provider's consistency over time in offering both functionality and conformance qualities, and can therefore be measured through the other two types of qualities over time. It can also be viewed as the 'goodwill' of a particular service provider. For example, Akamai [76] is generally considered as offering good quality services to end-users, due to its reputation and large market share developed in course of time.

While it is possible to consider all three types of quality for a service in the CDN peering environment, it is perhaps most interesting and relevant to address the issue of ensuring QoS from the view of quality as conformance. In this chapter, we adopt the conformance view and define QoS as the experience perceived by an end-user for being serviced by a CDN. We define quality in the following way [159]:

Let A be a CDN provider and $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$ *be the set of services provided by it. Assume that for each service* S_i, S_i^p *is the quality that A promised to offer to users and* S_i^d *is the actual delivered quality. The QoS for CDN A is,* $QoS_A = f(S_i^p, S_i^d)$*, where f is the function that measures the conformance between* S_i^p *and* S_i^d*.*

In this definition, we capture the notion of conformance generally, but do not specify how S_i^p and S_i^d can be measured. We anticipate measuring the QoS assessment of a CDN in terms of the *response time* perceived by an end-user for getting service from the provider. We also state that the QoS measure may take into account not only the average response time, but also the percentile (95^{th} percentile, for example) of the response time. Response time is a summation of processing and waiting time. Given

that fact that most Web objects are small, for which the processing time is negligible, we specify response time as the time a service takes to respond to various types of requests, i.e. expected waiting time for a request to be served (Section 4.2).

4.1.1 SLAs to Ensure QoS

Ensuring QoS guarantees requires a means of establishing a set of common quality parameters and establishing which attributes are needed by a particular customer to describe its QoS requirements. These factors are combined in an SLA that both a customer and provider agree to and that the provider refers to when monitoring its QoS performance. Examples of QoS parameters that an SLA may specify are:

- The maximum response time for a service request will not exceed 0.5 seconds.
- 95% of user requests will be completed in less than 2 seconds.
- A service will be available for at least 99.9% of time.

From a service management and business perspective, the fulfillment and assurance of SLAs is of key importance. In our context, the importance of an SLA lies in its encoding of performance obligations, so that a CDN can manage its workload. For example, the existing SLAs that a CDN currently holds firstly permit it to determine if a new SLA can be accepted as it is. Secondly, the SLAs are used to determine if the CDN is meeting user QoS requirements. Thirdly, a CDN uses its SLAs to quantify what further resources it would require in a peering arrangement. Thus, the existing SLAs are critical in establishing the runtime performance metrics for the CDN and as a basis to form CDN peering. Examples of attributes that an SLA encodes are:

- ***Service type.*** Content and/or application delivery requested by a user.
- ***Service requirements.*** Processing and capacity requirements to serve requests.
- ***Duration.*** The maximum time duration to serve content requests.
- ***Guaranteed QoS level.*** The level of guarantee (in terms of response time) that requests will be served within a delay threshold.

With this overview on QoS and SLAs for CDN peering, in the next section, we focus on developing performance models to improve end-user perceived experience.

4.2 Performance Models

An analytical model for CDN peering can be based on a complex combination of attributes such as Web server responsiveness, incoming traffic load, expected network delay, or geographic location. Several of these potential attributes vary over time and there is no single repository for listing the value of attributes such as geographic location or expected delay for all Internet-connected systems. Therefore, the values used in a CDN peering model are likely to be based on heuristics. In this section, we first position our effort with respect to other related work and develop simplified performance models based on queuing theory. For a given workload, mean and service time, we derive necessary expressions for measuring expected waiting time and cumulative distribution to measure a CDN's QoS performance.

4.2.1 Motivation and Scope

Several studies have focused on alleviating rapid and unpredictable spikes in request traffic. Many of these research efforts [53, 94, 105, 112, 125, 228] improve the greater good of the entire system for resource sharing and management. Recent proposals [23, 149, 200] also solve this problem using Peer-to-Peer (P2P) techniques. In CDNs context, despite the existence of a few research to model the behavior of CDNs, not much work has been done to understand the performance gains that can be achieved through CDN peering. Many of the prior work made simplifying assumption, thus losing generality. Moreover, most of the previous work analytically model a CDN as an M/M/1 queue, thus limiting the use of different service distributions for participating providers in a CDN peering arrangement.

Molina et al. [138] present a general expression for a CDN environment by following an M/M/1 queuing approach based on the assumption of Agrawal et al. [3] and study the performance impact of several design variables. Villela and Rubenstein [218, 219] introduce M/G/k/k queuing models for a collective of content providers and demonstrate a thresholding approach to improve performance of heavily overloaded systems. Ciciani et al. [58] model a CDN system as a multi-class queuing network by abstracting the whole network as a single finite Markov chain. Other

studies have mainly focused on server placement [37] or the evaluation of response time. Different techniques such as queuing theory-based modeling of Web server processing time [3, 36] and water filling schema [134] have been used for the response time evaluation. We differ by devising analytical performance models with the aim to demonstrate peering effectiveness and make concrete QoS guarantees for participating CDN providers in a peering arrangement.

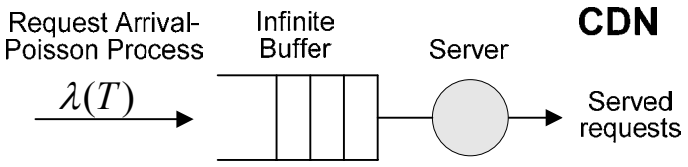


Figure 4.1: A CDN modeled as an M/G/1 queue.

4.2.2 Single CDN Model

Based on the observations that Web user arrivals follow a memoryless process with a constant arrival rate over a significant period of time [225], we model a CDN as an M/G/1 queue as shown in Figure 4.1. Table 4.1 shows the parameters and expressions that are used in the modeling.

Table 4.1: Parameter and expressions for the analytical model.

Parameter	Expression
Mean arrival rate	$\lambda = (1/T)$ (requests/second)
Mean arrival time	T
Mean service rate	μ
Load	$\rho = (\lambda/\mu)$
PDF of service distribution	$f(x) = \frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}} x^{-\alpha-1}, k \le x \le p$
Task size variation	α
Smallest possible task size	k
Largest possible task size	p
Expected waiting time	$E[W]$
Mean service time	$E[X]$

We assume that the total processing of Web servers of a CDN is accumulated through the server. The request streams coming to the Web servers of the CDN are abstracted as a single request stream. As we follow an M/G/1 queuing system, it allows us to define the service times independent of interarrival times and of one

another, and to have a general Probability Distribution Function (PDF). User requests arrive following a Poisson process with the mean arrival rate λ. All requests in its queue are served on a First-Come-First-Serve (FCFS) basis with mean service rate μ following a general distribution. Poisson modeling of end-user arrivals provides us a good basis for theoretical constructs and mathematical tractability. It also allows considering the complex network traffic dynamics as a "black box" and helps to estimate parameters (inputs) that are difficult to specify, collect, or measure in practice.

The term 'task' is used as a generalization of a request arrival for service. We denote the processing requirements of an arrival as 'task size'. We use the terms task and request arrival interchangeably. A request can be an end-user requesting an individual file or object, a Web page (containing multiple objects), the results of execution of a script (e.g. CGI, PHP) or any digital content. In a CDN, end-users request for content of varying sizes (ranging from small to large). Based on size of the content requested, the processing requirements, i.e. task size, also vary.

It has been found that Internet workloads are heavy-tailed in nature [68, 69], which can be characterized by the function, $\Pr\{X > x\} \sim x^{-\alpha}$, where $0 \le \alpha \le 2$. Based on this finding, we model the task size on a given CDN's service capacity to follow a Bounded Pareto distribution. The PDF for the Bounded Pareto $B(k, p, \alpha)$ service distribution is,

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}} x^{-\alpha-1}$$

where α represents the task size variation, k is the smallest possible task size, and p is the largest possible task ($k \le x \le p$). By varying the value of α, we can observe distributions that exhibit moderate ($\alpha \approx 2$) to high variability ($\alpha \approx 1$).

We start with the derivation of the expected waiting time $E[W]$, W is the time a user has to wait for service. $E[N_q]$ is the number of waiting customers and $E[X]$ is the mean service time. By Little's law, the mean queue length $E[N_q]$ can be expressed in terms of the waiting time. Therefore, $E[N_q] = \lambda E[W]$ and load on the server is, $\rho = \lambda E[X]$. Let $E[X^j]$ be the j-th moment of the service distribution of *tasks*. We have,

$$E[X^j] = \begin{cases} \dfrac{\alpha p^j ((k/p)^\alpha - (k/p)^j)}{(j-\alpha)(1-(k/p)^\alpha)} & \text{if } j \neq \alpha \\ \dfrac{\alpha k^\alpha \ln(p/k)}{(1-(k/p)^\alpha)} & \text{if } j = \alpha \end{cases}$$

By applying Pollaczek-Khintchine formula, we get the measure of expected waiting time, $E[W] = \lambda E[X^2]/2(1-\rho)$, with respect to different server load and task sizes.

Hyper-exponential approximation. To quantify the performance perceived by end-users while being serviced by a CDN, we need to find the PDF of waiting time distribution. The Bounded Pareto distribution has all moments finite; however, advanced analysis is complex due to the difficulties in manipulating the Laplace transforms of the queuing metrics, e.g. waiting time and busy period. Hence, the 'heavy-tailed' Bounded Pareto distribution can be approximated with a series of exponential distributions (known as Hyper-exponential), while still maintaining the main characteristics of the original distribution, such as heavy tail, first and second moments [30]. Therefore, we use an n-part Hyper-exponential service distribution that has the following PDF:

$$h_n(t) = \sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i t} \text{ , where } \sum_{i=1}^{n} P_i = 1$$

Service distribution and waiting time. We devise the PDF of the waiting time based on the above approximation of the service distribution. For that we start with finding the Laplace transform of the service distribution, $h_n(t)$, as follows:

$$L_{h_n}(s) = \int_0^\infty e^{-st} h_n(t) dt = \sum_{i=1}^{n} \frac{P_i \lambda_i}{\lambda_i + s}$$

The moments of the service distribution can be obtained as:

$$E[X^m] = (-1)^m \frac{d^m L_{h_n}(s)}{ds^m}\bigg|_{s=0}$$

where X is a continuous random variable with PDF $h_n(t)$. The first moment (mean) $E[X]$ and the second moment $E[X^2]$ are:

$$E[X] = \sum_{i=1}^{n} \frac{P_i}{\lambda_i} \quad \text{and} \quad E[X^2] = \sum_{i=1}^{n} \frac{2P_i}{\lambda_i^2}$$

The Laplace transform of the waiting time, $L_W(s)$ for an M/G/1 queue with the hyper-exponential approximation of a Bounded Pareto distribution is defined as:

$$L_W(s) = \frac{s(1-\rho)}{s - \lambda + \lambda L_{h_n}(s)} = \frac{s(1-\rho)}{s - \lambda + \lambda \sum_{i=1}^{n} \frac{P_i \lambda_i}{\lambda_i + s}}$$

This result can be numerically inverted to obtain the PDF of the waiting time distribution, $w(t)$, and the Cumulative Distribution Function (CDF), $W(t)$:

$$W(t) = \Pr[T \le t] = \int_0^t w(t)dt$$

Using the CDF, concrete QoS guarantees can be made regarding the waiting time experienced by a certain percentage of user requests. Figure 4.2 shows the CDF of waiting time of a CDN for the system load ρ = 0.5. Here, the hyper-exponential approximation of a Bounded Pareto distribution is used with α = 1.5, k = 1010.15 and p = 10^{10}. As for instance, there is about 50% probability that the waiting time experienced by end-users will be less than 20000 time units. Thus, a particular CDN can make concrete guarantees to provide at least this much QoS experience to its users.

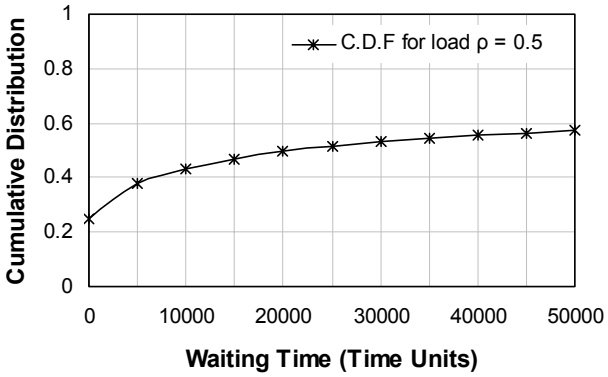


Figure 4.2: Cumulative distribution of waiting time.

4.2.3 CDN Peering Model

A CDN's inability to meet end-user QoS requirements according to the established SLAs lead to the formation of CDN peering, which assists the primary CDN to redi-

rect excess requests to the Web servers of its peers. Figure 4.3 provides a conceptual view of CDN peering where each CDN is modeled as an M/G/1 queue with general service distributions. It is abstracted that N independent streams of end-user requests arrive at a conceptual entity, called *dispatcher*, following a Poisson process with the mean arrival rate λ_i, $i \in \{1,2,\ldots,N\}$. The dispatcher acts as a centralized scheduler in a particular peering relationship with an independent mechanism to distribute requests among partnering CDNs in a user transparent manner. If, on arrival, a request can not be serviced by CDN i, it may redirect excess requests to the peers. Since the dispatching acts on individual requests of Web content, it endeavors to achieve a fine grain control level. We anticipate that it can also pave the ways in performing the *request assignment* and *redirection* at multiple levels—at the DNS, at the gateways to local clusters and also (redirection) between servers in a cluster. The dispatcher follows a certain policy that assists to assign a fraction of requests of CDN i to CDN j. The request stream to a CDN in the CDN peering model is $\lambda_{j,i}$ = request to CDN j for CDN i's content. For example, request to CDN B for CDN A's content is denoted as $\lambda_{B,A}$. For $\forall j \neq i$, $\lambda_{j,i}$ denotes redirected user requests, where CDN i is the primary where CDN j is a peer. For $\forall j = i$, $\lambda_{j,i}$ denotes the user requests to a primary CDN i.

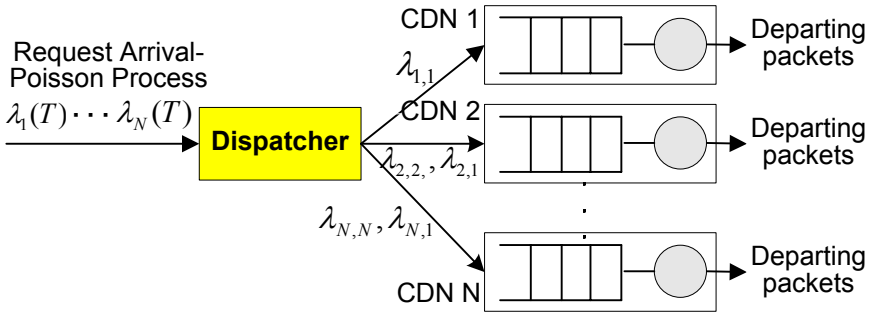


Figure 4.3: Conceptual view of CDN peering.

Inside each request stream, there is a FCFS service. Each stream is assigned a priority. Here, $p = 1,2,\ldots,P$ priority classes of end-user requests are assumed. A peer always prioritizes the requests from the primary CDN over its own user requests. However, if a redirected request (higher priority) arrives to a peer when its own user request (lower priority) is being served, it does not interrupt the current service. Thus, this priority discipline is non-preemptive during service quantum of user requests.

Waiting time. The classical result [59] on non-preemptive head-of-the-line (HOL) priority queue can be used to find the expected waiting time for the p-th (p = 1,2,…,P) priority user request,

$$W_p = \frac{W_0}{(1-\sigma_p)(1-\sigma_{p+1})}, \text{ where } \sigma_p = \sum_{i=p}^{P} \rho_i \quad \textbf{(4.1)}$$

W_0 is the average delay for serving a particular priority end-user request due to other requests found in service. It can be expressed as,

$$W_0 = \sum_{i=1}^{P} \frac{\rho_i E[X_i^2]}{2}$$

where $E[X_i^2]$ is the second moment of service time for a user from class i. It can be mentioned that W_p does not depend on user requests from lower priority class, i.e. i = 1, 2,…, p-1, except for their contribution to the numerator W_0 [115].

Let us assume that the user requests for the primary CDN belongs to the p-th priority class. The Laplace transform of the waiting time for the primary CDN is denoted as $W^*_p(s)$. Using the known solution [66] for the distribution of waiting time for each priority group in a priority queue, it can be expressed as,

$$W_p^*(s) = \frac{(1-\rho)s + \lambda_L[1 - \sum_{j=1}^{p-1} \frac{\lambda_{j,j}}{\lambda_L} B_j^*(s)]}{s - \lambda_{p,p} + \lambda_{p,p} B_p^*(s)}, \text{ where } \lambda_L = \sum_{j=1}^{p-1} \lambda_{j,j} \quad \textbf{(4.2)}$$

Similarly, for any peer with the priority in the range *1,2 ...,(p-1)* the Laplace transform of waiting time is found as,

$$W_j^*(s) = \frac{(1-\rho)[s + \lambda_{p,p} - \lambda_{p,p} G_p^*(s)]}{s - \sum_{j=1}^{p-1} \lambda_{j,j} + \sum_{j=1}^{p-1} \lambda_{j,j} \sum_{j=1}^{p-1} B_j^*(s + \lambda_{p,p} - \lambda_{p,p} G_p^*(s))} \quad \textbf{(4.3)}$$

Here, $G^*_p(s)$ is the transform for the M/G/1 busy period distribution for a p-priority class, which is expressed as,

$$G_p^*(s) = B_p^*(s + \lambda_{p,p} - \lambda_{p,p} G_p^*(s)) \quad \textbf{(4.4)}$$

QoS performance. The ability to gauge the QoS of a CDN provider is crucial for achieving effective service from it. The PDF of the waiting time distribution (through numerical inversion) for each CDN, with independent priority class can be used to observe the expected waiting time perceived by the majority of end-users in the CDN peering system. Since a primary CDN's request has priority over any peer's own user requests, we can consider using (4.2) for a primary CDN, while (4.3) for any peer. Although these equations are useful for computation, the iterative expression for $G^*_i(s)$ in (4.4) is impossible to invert numerically [115]. Therefore, in our work, the waiting time experienced by a primary CDN's user requests is found using (4.2), while the classical result presented in (4.1) is used to find the average expected waiting time for a peer's end-user requests.

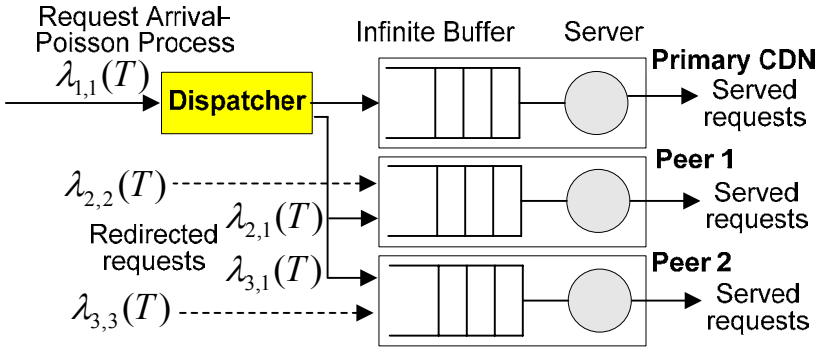


Figure 4.4: A peering scenario with three CDNs.

4.3 Performance Results

We now present the performance results obtained for the analytical models. We use Mathematica[2] to simulate a CDN peering arrangement consisting of three CDNs, as shown in Figure 4.4. Table 4.2 lists the notations used for this scenario.

The scenario of Figure 4.4 provides a snapshot of the system when CDN 1 is in the role of a primary, while CDN 2 and CDN 3 are acting as peers. However, these roles may change over time, as the roles are dynamic and interchangeable depending on load (Chapter 3: Section 3.1). Each CDN is modeled as an M/G/1 queue with

[2] An integrated computational software program that is used in scientific, engineering, and mathematical fields and other areas of technical computing. Please check: http://www.wolfram.com

highly variable Hyper-exponential distribution that approximates a heavy-tailed Bounded Pareto service distribution (α, k, p) with variable task sizes. Thus, the workload model incorporates the high variability and self-similar nature of Web access. Table 4.3 shows the characteristics of the workload model. For our experiments, we consider the expected waiting time as an important parameter to evaluate the performance of a CDN. In the peering scenario, we assume that all peers hold the content required to serve redirected requests from the primary and there exists an SLA for serving all user requests by the primary CDN in less than 20000 time units.

Table 4.2: List of notations used in the CDN peering scenario.

Notation	Description
N	Number of CDNs, $N = \{1,2,\ldots,N\}$
$E[X_i^j]$	j-th moment of CDN i's service distribution
ρ_i	Initial load on CDN i, $\rho_i = \lambda_{i,i} E[X_i]$
ρ_i^*	New load on CDN i, $\rho_i^* = \lambda^*_{i,i} E[X_i]$
$\lambda_{i,i}$	Initial arrival rate at CDN i
$\lambda^*_{i,i}$	New arrival rate at CDN i, $\lambda^*_{i,i} = \lambda_{i,i} - \lambda^i_{redirect}$
$E[W_i]$	Expected waiting time (initial) at primary CDN i,
$E[W_i^*]$	Weighted average expected waiting time at CDN i, $E[W_i^*] = \frac{\rho - \rho^i_{redirect}}{\rho} E[W_i] + \sum_{k=1}^{i-1} \frac{w_k \cdot \rho^i_{redirect}}{\rho} E[W_k]$ w_k depends on the redirected ratio to a given peer
$\rho^i_{redirect}$	Fraction of content requests redirected from CDN i

Table 4.3: Workload model.

Category	Distribution	P.D.F	Range	Parameters
Primary CDN, $0.1 \le \rho \le 0.9$	Hyper-exponential	$h_n(t) = \sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i t}$	$x \ge k$	$\alpha = 1.5$, $k = 1010.15$, $p = 10^{10}$
Peer 1, $\rho = 0.5$	Hyper-exponential	$h_n(t) = \sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i t}$	$x \ge k$	$\alpha = 1.5$, $k = 1010.15$, $p = 10^{10}$
Peer 2, $\rho = 0.4$	Hyper-exponential	$h_n(t) = \sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i t}$	$x \ge k$	$\alpha = 2$, $k = 1500.23$, $p = 10^{10}$

4.3.1 QoS Performance of the Primary CDN

First, we provide the evidence that CDN peering is able to assist a primary CDN to provide better QoS to its end-users. The CDF of the waiting time of the primary CDN can be used as the QoS performance metric. In a highly variable system such as a

CDN peering arrangement it is more significant than average values. The waiting time corresponds to the time elapsed by a user request before being serviced by the CDN. Figure 4.5 shows the CDF of waiting time of the primary without peering for different loads. We observe that for a fair load ρ = 0.6 there is about 55% probability that users will have a waiting time less 20000 time units (SLA constraint). For a moderate load ρ = 0.7, there is about 50% probability for users to have waiting time below the threshold, while for a heavy load ρ = 0.9 it reduces to about > 24%.

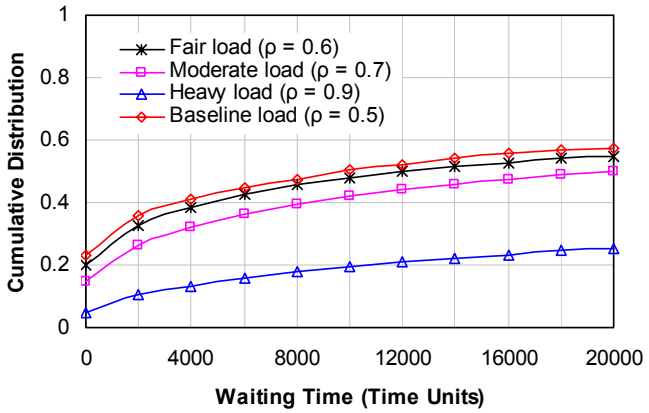


Figure 4.5: Cumulative distribution of waiting time (primary) without peering.

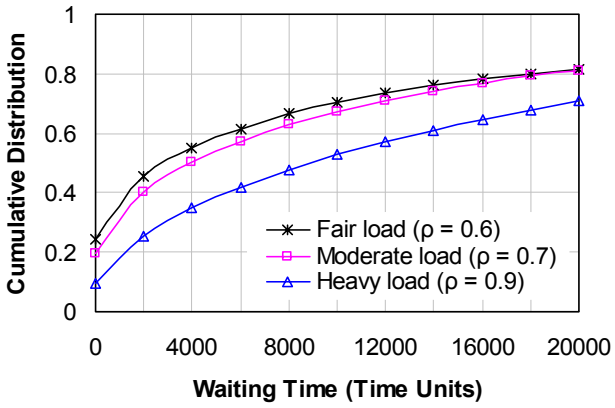


Figure 4.6: Cumulative distribution of waiting time (primary) in peering.

The CDN peering model arranges the participating providers according to a non-preemptive HOL priority queuing system (Section 4.2.3). It is an M/G/1 queuing system in which we assume that user priority is known upon their arrival to a CDN and therefore they may be ordered in the queue immediately upon entry. Therefore, various priority classes receive different grades of service and requests are

discriminated on the basis of *known* priority. Thus, in our model an incoming request (with priority p) joins the queue behind all other user requests with priorities less than or equal to p and in front of all the user requests with priority greater than p. Due to this nature of the CDN peering model, the effect of peering can be captured irrespective of any particular request-redirection policy. We provide necessary result to support this claim in Section 4.3.3.

Figure 4.6 shows the C.D.F of the primary CDN with peering for different loads. By comparing Figure 4.5 and Figure 4.6, it can be found that for a fair load $\rho = 0.6$ there is about 80% probability that users will have a waiting time of less than the SLA constraint, 20000 time units. Therefore, peering assists the primary to achieve a QoS performance improvement of about 31%. For a moderate load $\rho = 0.7$, there is > 81% probability for end-users to have waiting time below the threshold, an improvement of about 38%. For a heavily loaded primary CDN with $\rho = 0.9$ the probability becomes about 70%, which lead to an improvement of > 65%. Moreover, for loads $\rho >$ 0.9, still higher improvement can be predicted. Based on these observations, we state that CDN peering achieves substantial QoS performance improvement when compared to the non-peering case.

4.3.2 Request-Redirection Policies

Now we focus on using request-redirection within the CDN peering model. We assume no redirection until the primary CDN's load reaches a *threshold* load ($\rho = 0.5$). This load value is also used as the *baseline* load for comparing waiting times at different primary CDN loads. Any load above that will be 'shed' to peers. A request-redirection policy determines which requests have to be redirected to the peers. Each peer is ready to accept only a certain fraction (*acceptance threshold* < 50%) of the redirected requests. Redirected requests to a given peer exceeding this acceptance threshold are simply rejected for service to maintain the system equilibrium. In the face of sudden surge in demand, the load on a given primary CDN i, where $i \in \{1,2,\ldots,N\}$ becomes, $\rho^*_i = \rho_i - \rho^i_{redirect}$ and the redirected load is distributed among the peers. The value of $\rho^i_{redirect}$ varies depending on the dispatcher-chosen redirection policy. The initial and new load on a primary CDN i is measured by, $\rho_i = \lambda_{i,i} E[X_i]$ and

$\rho^*_i = \lambda^*_{i,i} E[X_i]$ respectively. Here, $\lambda_{i,i}$ is the initial request arrival rate, whereas $\lambda^*_{i,i} = \lambda_{i,i} - \lambda^i_{redirect}$ is the new request arrival rate after redirection.

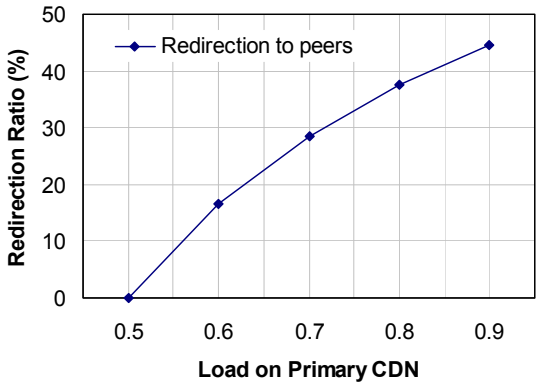


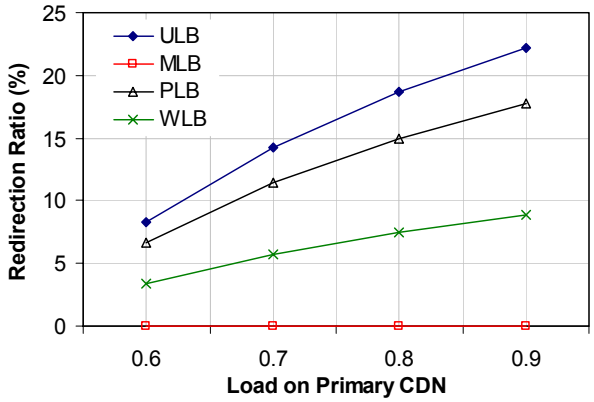
Figure 4.7: Redirection ratio at different primary CDN loads.

We define the following request-redirection policies that are used by the dispatcher in the CDN peering model to distribute redirected requests to peers:

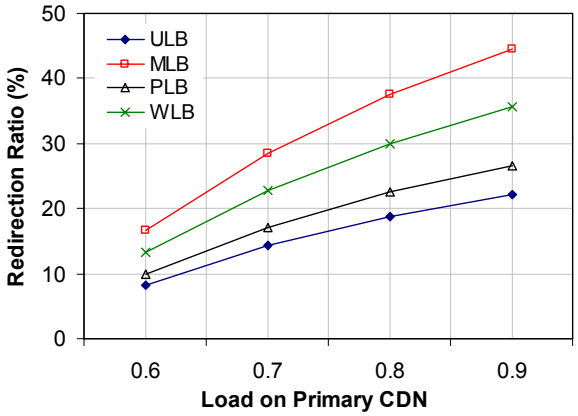
- ***Uniform Load Balanced (ULB).*** It distributes the redirected content requests uniformly among all the peers.
- ***Minimum Load Balanced (MLB).*** It assigns the redirected content requests to the peer with minimum expected waiting time.
- ***Probabilistic Load Balanced (PLB).*** It distributes redirected content requests to the peers according to a certain probability. This probability depends on the load threshold for the fraction of redirected requests that a peer can accept from the primary. In the peering scenario (Figure 4.4), we use probabilities of 0.4 and 0.6 to assign a certain fraction of the redirected requests to peer 1 and peer 2 (i.e. CDN 1 and CDN 2), respectively. We measure this probability based on the acceptance threshold and the service capacity of the peers.
- ***Weighted Load Balanced (WLB).*** It assigns 80% of redirected content requests to the peer with minimum expected waiting time. The remaining 20% of traffic is uniformly distributed over all other peers.

The amount of redirected requests is denoted as the *redirection ratio*, which is quantified as a percentage of the primary CDN's load. The influence of different

primary CDN loads on the redirection ratio is shown in Figure 4.7. We find that the redirection ratio increases with the growing primary CDN's load, independent of any particular request-redirection policy. In Figure 4.8, the redirection ratios assigned to the peers are shown. Each curve denotes a different redirection policy and x-axis denotes the load on the primary CDN.



(a) Peer 1



(b) Peer 2

Figure 4.8: Distribution of redirected requests to peers.

In the ULB request-redirection policy, both peer 1 and peer 2 receive the same amount of redirected requests from the primary. When the MLB request-redirection policy is used by the dispatcher, it assigns all the redirected requests to peer 2, which has the minimum expected waiting time. Hence, no redirected request is assigned to peer 1. If PLB request-redirection policy is used by the dispatcher, it leads to a distribution of 40%-60% of the redirected requests to peer 1 and peer 2 respectively. A

dispatcher following the WLB request-redirection policy assigns 80% of redirected requests to peer 2 (with minimum expected waiting time) and the rest 20% to peer 1.

4.3.3 Impact of Request-Redirection

Now we study the impact of request-redirection on the expected waiting time of end-users on the primary CDN. Without request-redirection when the primary CDN's load approaches to 1.0, the user perceived performance (in terms of waiting time and queue length) for service by the primary CDN tends to infinity. On the other hand, with request-redirection the waiting time of the primary CDN decreases as the requests are redirected to the peers.

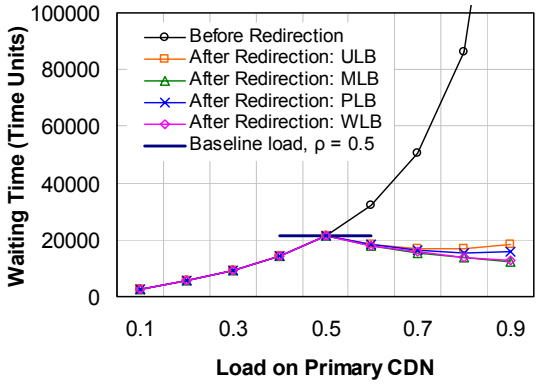


Figure 4.9: Impact of request-redirection on waiting time of the primary CDN.

Typically redirecting excess requests improves performance on the primary CDN. Figure 4.9 provides this evidence by showing the performance improvement (in terms of waiting time) that the primary gains for all the request-redirection policies. Here, we compare the expected waiting time as a function of system load under four request-redirection policies, by considering lightly loaded peers (load of peer 1 and peer 2 are set to ρ = 0.5 and ρ = 0.4 respectively), while tuning the primary CDN's load ($0.1 \leq \rho \leq 0.9$). It can be noted that a weighted average value of waiting time is presented in order to capture the effect of request-redirection. From the figure it can be seen that as more requests are redirected to the peers, they realize higher waiting time due to the peers' own load.

Table 4.4 summarizes the reduction of the expected waiting time for the primary CDN in peering for different request-redirection policies. For all the four policies, it is

observed that substantial performance improvement is achieved on the expected waiting time when compared to the non-peering case. Among all the four policies, ULB, PLB and WLB effectively have *redirection ratio* as the common performance parameter. For all these three policies, redirected requests are distributed among peers according to a certain percentage. Therefore, to some extent they exhibit similar characteristics. MLB assigns all redirected requests to a single peer. Although results for MLB may show as good performance as the other three policies (due to light load and less heavy-tail workload on peer 2), there is a possible concern for the peer with minimum expected waiting time to become overloaded with the redirected requests (herd effect [70]). Therefore, it is preferable to spread the load of redirected requests among multiple CDNs rather than assigning all redirected requests to a single peer.

Table 4.4: Reduction of waiting time on the primary CDN.

Load on primary CDN	**Reduction in waiting time %**			
	ULB	**MLB**	**PLB**	**WLB**
Fair load, $\rho = 0.6$	43.20%	44.41%	43.66%	44.24%
Moderate load, $\rho = 0.7$	66.31%	69.31%	67.50%	68.91%
Heavy load, $\rho = 0.9$	90.52%	93.70%	91.94%	93.39%

From the results, it is clear that all the request-redirection policies guarantee that the maximum waiting time is below the SLA constraint of 20000 time units. This confirms that redirecting only a certain fraction of requests reduces instability and overload in the system because the peers are not overwhelmed by bursts of additional requests.

4.3.4 Measurement Errors

The dispatcher bases its redirection decision on the measured value of the primary CDN's load. So far we have assumed that perfect information is available for this decision. However, the dispatcher can have inaccurate information about the load on the primary CDN, e.g. due to delays in receiving the measurements. Therefore, the impact of measurement errors on the effectiveness of the redirection policies can be measured. Let us denote the measured load of the primary CDN at the dispatcher by $\tilde{\rho} = \tilde{\lambda} E[X]$, with $\tilde{\lambda} = \lambda(1 \pm \varepsilon)$, where ε is the percentage of the correct load ρ.

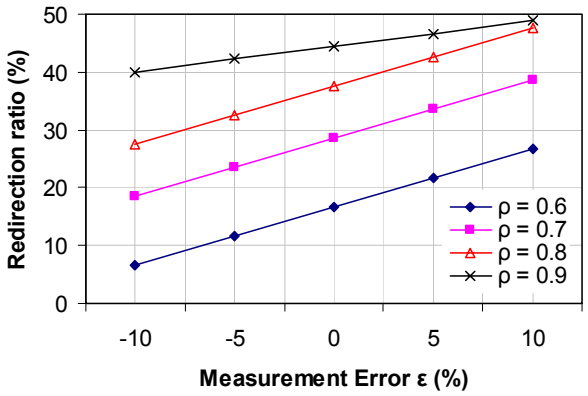
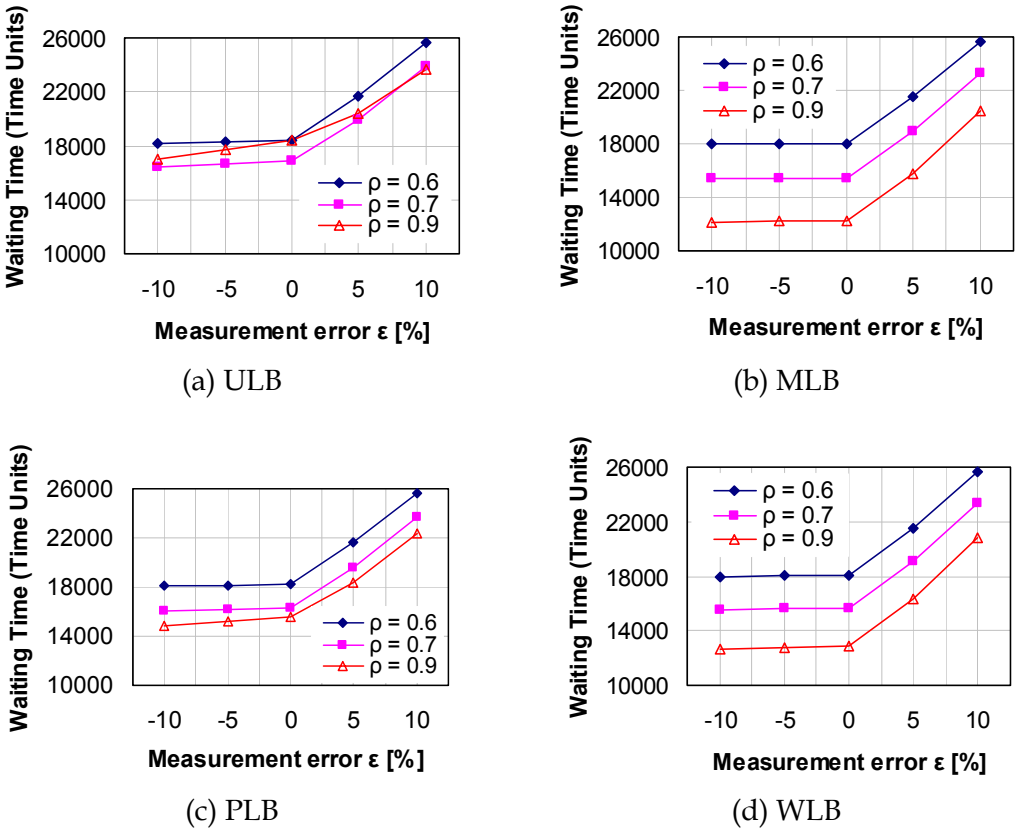


Figure 4.10: Impact of measurement errors on redirection ratio.



(a) ULB (b) MLB

(c) PLB (d) WLB

Figure 4.11: Waiting time variation for different load measurement error.

Figure 4.10 shows the effects of primary CDN's load measurement error on the redirection ratio. Each line denotes a different primary CDN load ρ, and the x-axis denotes the measurement error ε, in percentage of ρ. A value of 0 on the x-axis corresponds to perfect primary CDN load information. We have observed that the redirection ratio changes more for positive ε than for the corresponding negative ε.

Figure 4.11 shows the impact of primary CDN's load measurement error on the waiting time for different request-redirection policies. It can be observed that in all the four cases, for measurement error $\varepsilon > 0$, the dispatcher assumes the primary CDN's load to be higher than what it is and hence it redirects more requests than the actual load. The extra redirections incur additional waiting time for the user requests and hence it increases linearly from $\varepsilon = 0$. For negative ε, the dispatcher assumes the primary CDN's load to be less than the actual and hence redirects pessimistically. As a result, requests on the primary could experience greater expected waiting time for being processed. However, the average of waiting time normalizes the adverse effect in order to keep the performance at an acceptable level. Nevertheless it can be concluded that greater accuracy is needed in load measurement of the primary CDN.

4.4 Summary and Conclusion

In this chapter, we have presented innovative analytical models for CDN peering. According to the model, an overloaded CDN is stabilized by offloading a fraction of the incoming requests to its peers. We have also demonstrated the effects of peering and predicted end-user perceived performance from a given CDN. We have shown that it is easier to meet QoS goals through CDN peering, and that any resulting system is more resilient to flash or periodic crowds.

Although the performance models are simplified in order to accommodate the system complexities, we believe that they provide a foundation for performing effective peering between CDNs though achieving target QoS in service delivery to end-users. Our model-based approach is important since having each CDN provider communicate how it would service millions of potential end-users would introduce significant scalability issues, and requesting this information from each partnering provider at the user requests time would introduce substantial delays.

Based on the findings from this chapter, we set to investigate techniques to enable CDN peering. Specifically, in the next chapter, we develop distributed load sharing strategy using dynamic request-redirection and provide discrete-event simulation analysis of the proposed redirection mechanism.

Chapter 5

Resource Discovery and Request-Redirection

A dominant factor for the success of CDN peering is to serve each end-user by an optimal Web server, in terms of network cost, even under heavy load conditions. Before it can be comprehended, appropriate resource discovery and request-redirection mechanisms, coupled with an optimal server selection strategy, should be in place to perform the distribution of highly skewed loads. In this chapter, we devise a load distribution strategy by adopting *distributed* resource discovery and *dynamic* request-redirection mechanisms, taking traffic load and network proximity into account. We exercise an asynchronous resource discovery protocol, reminiscent of the publish/subscribe notion, and formulate the resulting redirection scheme. We provide extensive simulation analysis to show that our approach is effective to handle heavy loads by preserving locality, and thus achieve service "responsiveness".

5.1 Introduction

The success of peering and the effectiveness of operations for content delivery in a CDN peering system depend on its ability to perform *resource discovery*, *server selection* and *dynamic request-redirection* under degenerated load conditions, e.g. flash crowds. The resource discovery process specifies how external resources offered by disparate CDNs are discovered. An effective server selection strategy determines the least loaded edge server(s) that is best suited to serve user requests. The server selection phase typically chooses the "nearest" optimal server to the requesting user. A

dynamic request-redirection mechanism assists in directing user requests to the target edge server(s), in order to alleviate imbalanced load situations. These phases may be interleaved to collectively perform load distribution by reacting to overload conditions in a multi-provider CDN peering system and thus endeavor to achieve scalability. Specifically, load distribution in our context can be stated as:

> *Given current load information in an overload condition, the CDN peering system needs to select an optimal server to which a given request can be redirected. To obtain up-to-date load information, methods for load monitoring and load index dissemination is necessary.*

In an effective load distribution strategy, a load index of resources needs to be estimated with low computation and communication overhead. In addition, network proximity information (distance between users and servers) is also important for load distribution decision so that requests are directed towards nearby servers. Therefore, an ideal load distribution strategy should realize a redirection scheme that takes traffic load and proximity into account. In our approach, load indices of distributed inter-CDN servers are obtained through an *asynchronous feedback mechanism* and network proximity is measured using a *pinger logic* with low messaging overhead.

The aims of this chapter are: i) to perform dynamic load distribution under traffic surges by redirecting excess requests to the least loaded Web server(s), thus binding users to optimal replicas (*timeliness*); (ii) to exhibit acceptable throughput under overload conditions (e.g. during *flash crowds*); iii) to scale to distributed inter-CDN resources scattered across the globe (*dynamic lookup*); and iv) to maintain administrative control over local resources and their states (*resource encapsulation*).

5.2 Motivation and Scope

Resource discovery is a popular topic in large-scale distributed systems; whereas, request-redirection is an indispensable enabling cornerstone for CDNs. Many previous research efforts [44, 46, 62, 65, 97, 120, 190, 194, 230] have focused on these two topics separately in different domains, such as Grid computing, P2P-based systems, multi-agent systems, distributed Web servers, Internet, overlay and ad-hoc networks. However, they can not be directly applied for load distribution in a CDN peering

system, due to the necessity for handling dynamic circumstances, thus requiring up-to-date information about widely-distributed resources. In addition, providers should learn about available resources quickly, without using an inordinate amount of communication, and the resource discovery and redirection algorithms may be used repeatedly to obtain updated resource status information. There are also other challenges, which include virtualization of multiple providers and offloading requests from the overloaded provider to its underloaded peers, based on cost, performance, and load. In such a cooperative multi-provider environment, requests are directed to sets of servers deployed across multiple CDNs as opposed to individual servers belonging to a single entity. Therefore, resource discovery and request-redirections must occur over distributed sets of servers spanning multiple CDNs, without having complete state information.

Analysis of previous research efforts, in relation to CDN peering, suggests that there has only been modest progress on devising resource discovery and request-redirection mechanisms. We justify this observation by comparing our approach with existing schemes. As mentioned previously (Chapter 3, Section 3.2), CDN peering initiatives and related research such as Content Distribution Internetworking (CDI) model [72], request-routing mechanisms for content networks [19], CDN Brokering [19], modeling traffic redirection [9, 182], load distribution across geographically distributed Web servers [46] have mostly focused on describing peering architecture, performance evaluation, and redirection modeling. Many of them do not show the effectiveness of any particular redirection or load distribution mechanism. While we complement these research efforts, the CDI model in particular, we differ by devising mechanisms for resource discovery and request-redirection to perform dynamic load distribution.

Ercetin et al. [79] model request-redirection for CDN brokering as a delay-constrained routing problem. Unlike our work, the devised solution could only be applied to a snapshot of the system, thus limiting its scalability. Alzoubi et al. [8] present a load-aware IP Anycast CDN, incorporated with a route-controller, which takes server and network load into account to realize anycasting. This work estab-

lishes the applicability of anycasting as a redirection technique in CDNs. Our work is in line with them, as we focus on minimizing the cost associated with request-redirection. We differ by avoiding the use of a centralized route controller and by not following a post-processing approach to offload overloaded servers. Conti et al. [65] presents a QoS-based architecture for load distribution among replicated Web servers. While this approach might be effective to handle highly skewed loads, the focus is particularly on a single domain of clustered servers. Another notable work is a request distribution system for PlanetLab [194], which considers server loads and proximity information from pre-defined landmarks to route requests. It performs centralized load index dissemination and arbitrary server selections. On the contrary, we follow a decentralized approach to improve adaptability to dynamic changes.

There are also several representative work on resource discovery (and request redirection, to some extent) in large-scale distributed networks, i.e. non-CDN domains, such as Grids and P2P systems. Harchol-Balter et al. [97] present a distributed resource discovery algorithm, called name-Dropper, for large distributed networks of computers. This algorithm achieves near-optimal performance both with respect to time and network communication complexity. However, it may lead to the same particular target server selection by multiple originating machines. Lamnitchi et al. [120] study the resource discovery problem in a resource-sharing environment that combines the complexity of Grid and the dynamism of P2P networks. They propose a Grid emulator for evaluating resource discovery techniques based on request propagation. Unlike us, their work is targeted to a P2P-based Grid system, which does not realize CDN peering characteristics.

P2P networks support unstructured, e.g. Gnutella [199] and Kazaa [126], or structured, e.g. CAN [185], Chord [206], and Pastry [187], discovery services. In the first approach, hosts and resources are made available on the network without an overlay planning. The latter exploits highly structured overlays, using distributed indexing data structure, such as Distributed Hash Tables (DHTs) to route queries over a P2P network. The peer discovery and membership services are mainly used for the construction and startup of P2P networks. Nevertheless, with the presence of

mechanisms to virtualize multiple providers and realize redirection for dynamic load distribution, P2P techniques such as gossip algorithms [27] can be effective for resource discovery in CDN peering.

5.2.1 Resource Discovery

In the CDN peering system, an overload condition occurs when incoming load on the primary exceeds a given *alarm threshold*, as reported by its server(s). Under such circumstances, the conduit or gateway, i.e. PA, Mediator and PR as a single conceptual entity (Chapter 3: Section 3.3), of the primary discovers external resources.

The candidate resource discovery algorithm realizes a distributed nature, since there is no central control in the system. CDNs operate independently of each other; making local queries within its domain and transferring global information about part or all to the CDN peering system. A Service Registry (SR) instance in each CDN contains local resource information. Once a peer delegates its resources for use by the primary, gateways of the peers interact to register the offered resources to the global SR, which is a distributed SR implementation (Section 5.4.1), containing information of all resources in the peering arrangement. The list of registered resources in the global SR is updated periodically to realize a soft-state protocol for membership management. Gateways contact addresses are learned via out-of-band information.

The communication protocol to aid resource discovery is reminiscent of the publish/subscribe paradigm. This approach endeavors to perceive—scalability and full decoupling from other system operations; a possibly "offline" approach due to the asynchronous nature of resource discovery; and indirect addressing for load distribution. However, our approach differs from the publish/subscribe system in terms of functionality. While the latter deals with resource publishing and message dissemination, our main focus is on efficient resource discovery. In addition, a publish/subscribe system is intended to find all potential participants who are capable of serving user requests. In contrast, our redirection strategy, followed by the resource discovery, attempts to find an "optimal" peer, not all of them. As a consequence, a smaller fraction of the whole system is traversed and communicated.

5.2.2 Server Selection and Request-Redirection

An effective request-redirection mechanism to perform load distribution in a CDN peering system could be devised by examining two alternatives. In the first case, all peers belonging to a subCDN (Chapter 3: Section 3.1) could provide a real-time load status of each surrogate in the global subCDN using standard metrics. The primary CDN (or an authoritative entity belonging to it) can compare these load indices with its metrics and choose whether requests have to be redirected to a peer. Practical constraints could be put in place to ensure that redirection cost is minimized and peers' servers are not overloaded. Alternatively, an independent third party could supervise and manage all the peers for load distribution. However, security, trust, and proprietary issues make it unlikely that a CDN would agree to have an external party to make allocations of its resources according to some load distribution policy. Therefore, we follow the first approach by interleaving server selection and request-redirection to perform load distribution. Our approach seeks to prevent wide oscillation in the load distribution decisions by selecting optimal server(s) through cost minimization, taking traffic load and network proximity into account.

5.3 Algorithms

In this section, we first describe the workings of the resource discovery protocol for the CDN peering system. Then we present a load and proximity-aware redirection strategy, coupled with optimal server selection to perform dynamic load distribution.

5.3.1 Resource Discovery Formulation

Let M be the set of all possible resources from participating CDNs, with N users spreading across the system. Content request from user i is denoted as $r_i \in R$, which is a constraint on the set of available resources. Let $match(r_i) \in M$ be the set of servers that satisfy r_i. The primary CDN A, which receives the incoming request r_i is expected to provide the set of resources $match(r_i)$. However, under peak load, CDN A may not be able to serve r_i, and therefore, searches for other peers to serve the request on its behalf. Let us consider a peer B, which can provide resources $match(r_i')$, satisfying

user request r_i'. We say that r_i and r_i' overlap iff $match(r_i) \cap match(r_i') \neq \varnothing$ and there exists at least one resource that satisfies both r_i and r_i'. Therefore, least loaded servers from CDN *B* can be utilized by CDN *A* to satisfy the content request.

```
     begin ...
1:   for all r ∈ R do
2:     if alarm_flag = false && match(r) ≠ ∅ then
3:       Serve content request r from primary CDN's Web servers
4:     else if alarm_flag = true || match(r) = ∅ then
5:       if WSList = null then
6:         Populate n, WSList from p peers using lookupMethod
7:       end if
8:     end if
9:     for i =1 to n do
10:      Send ServiceRequest(r) to i.WSList
11:      Receive ServiceResponse(AcceptedReqList, RejectedReqList) from i.WSList
12:    end for
13:  end for
     ...
     end
```

Figure 5.1: Service Request from the primary CDN during resource discovery.

```
     begin ...
1:   for all s ∈ S do
2:     if s can be serviced && s is not already accepted then
3:       Add s to AcceptedReqList
4:     else if s can not be serviced then
5:       Add s to the RejectedReqList
6:     end if
7:   end for
8:   Send ServiceResponse(AcceptedReqList, RejectedReqList) to the primary CDN
     ...
     end
```

Figure 5.2: Service response from a peer's server during resource discovery.

Distributed resource discovery algorithm. Figure 5.1 and Figure 5.2 respectively present the pseudocode for service request and service response procedures, which are required during resource discovery. The principle of resource discovery in CDN peering is as follows. Upon receiving user requests for content, the primary CDN

finds suitable Web servers to serve them (Figure 5.1: Lines 1 to 3). In order to preserve locality, the primary CDN issues local queries to the local SR instance, to find the potential local resources that are able to serve the incoming requests. Under traffic surges, incoming load of a primary CDN exceeds a given *alarm threshold*. When this occurs, an *alarm_flag* is set to indicate that no optimal resource is found within the local domain. The primary uses a *lookupMethod* to contact peers and populate a list *WSList* of available resources from participating providers (Figure 5.1: Lines 4 to 8). The primary sends a *ServiceRequest* message to peer(s), containing the required service requirements, and receives the *ServiceResponse* from the peers' servers (Figure 5.1: Lines 9 to 13). When a peer's server receives a *ServiceRequest* message, it chooses the requests that it accepts to serve and the ones that it refuses to serve, based on whether it can meet the service requirements. It then sends a *ServiceResponse* message containing the lists of acceptable and rejected requests (Figure 5.2: Lines 1 to 8). Upon receiving peers' response, excess requests are redirected from the primary to the least loaded peer(s), using the redirection mechanism outlined in the next section.

The *lookupMethod* during service request determines the way peer selection is performed. It depends on whether the peers share partial information about their services, i.e. topology, connectivity information, dynamic link properties (link weight, background traffic, and congestion level), and dynamic node properties (bandwidth and processing power, shared by peers' servers). The *lookupMethod* may follow *broadcast* or *selective* mode. The first operational mode is the only choice when a CDN does not possess an SR instance to store its local resources information. With the presence of an advertising mechanism and the existence of an SR instance at each CDN, peer selection can be performed selectively. We exploit dynamic status and availability information of offered resources from peers, by searching the global SR. We preserve locality, since the attempt is to find local resources first, thus realizing *dynamic lookup* and increasing *resource encapsulation* within the primary's domain.

Other complementary selective modes could be: *controlled anycast*, *greedy random* and *greedy ordered*. Controlled anycast employs a route controller [212, 214] to which the Web servers within the CDN peering system advertise the anycast address. As

such, the route controller can influence the route selection to peers using some external intelligence [8]. In the greedy random mode, one peer is selected that can serve the content request on behalf of the primary, chosen uniformly at random. The greedy order mode chooses a peer according to some predefined criteria, such as network factors and QoS.

Table 5.1: Annotation of the resource discovery algorithm.

Properties	**Parameters**	**Description**
Methodology	Notion	Fully decoupled "offline" approach (*asynchronous*)
	Interconnection topology	CDN Peering (*system under consideration*)
Implementation	Granularity	End-user requested content
	Distribution	Distributed query-based approach
	Data integration scheme	Gateway (*information exchange and request-redirection*)
	Peer selection	Resource encapsulation and dynamic lookup (*searching within local domain first*)
Scalability	Data volume or index representativeness	Scalable content-based searching upon user requests
	Traffic intensity	Incoming traffic for replicated content
	Resource identification	Request mapping (*accepted and rejected requests lists*)

Table 5.1 summarizes the properties of the resource discovery algorithm. This annotation avails to analyze our approach and assists in making the right system design choice. The first property specifies the basic notion of resource discovery and the domain within which it is initiated. The next property focuses on the implementation, specifying the granularity—task unit that the system can support; distribution—particular design choice; data integration scheme—how to gain access to necessary data of interest; and peer selection—how locality is preserved for effective resource discovery. The last property delineates the perceived scalability of our approach by stating the prime means to support resource discovery, instrumentation of user access pattern, and identification of available resources.

5.3.2 Request-Redirection Formulation

Since request-redirection is at the heart of our load distribution strategy, we first formulate the redirection problem and then present the devised algorithm. We define

redirection cost R_c, for serving requests through redirection in overload conditions. It varies for different servers of peers, as it depends on a server's traffic load and network proximity (in terms of round-trip response time). Specifically, R_c is defined as:

$$R_c(i,j) = \begin{cases} \infty & \text{if } p_{ij} = \infty \\ l_{ij} p_{ij} & \text{otherwise} \end{cases}$$

where l_{ij} is the incoming traffic load from user i on server j, and p_{ij} is the network proximity between them. It is known that a server's response load indicates the potential load assigned to it, since request and response loads on a CDN server are linearly correlated [8]. Therefore, the request load l_{ij} (traffic volume) can be calculated based on the server load. A server j's load is the product $u_j S_j$, where u_j is the server utilization in [0, 1] as reported by the load monitoring apparatus (mediator and SR) and S_j is its capacity, specified by the maximum number of serviced requests/second as reported in the CDN server's configuration specifications.

The delay caused by inter-CDN redirection depends on the network proximity p_{ij}:

$$p_{ij} = \begin{cases} rt_{ij} & \text{if } j \text{ is an intra - CDN Web server} \\ rt_{ij} + I_D & \text{if } j \text{ is an inter - CDN Web server} \end{cases}$$

where rt_{ij} is the response time from user i to server j and I_D is the delay. To minimize redirection cost during load distribution, the redirection problem is formulated as:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} R_c(i,j) a_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{N} a_{ij} = 1, \forall j \\ & \sum l_{ij} a_{ij} \le S_j, \forall j \\ & a_{ij} \in \{0,1\}, \forall i, j \end{aligned}$$

where a_{ij} is an indicator variable to determine whether the Web server j is in the same CDN as user i; $a_{ij} = 1$ if server j is an inter-CDN server, and $a_{ij} = 0$ otherwise. Finding an optimal solution for the above Integer Linear Program formulation is NP-hard, when a_{ij} is an integer. Therefore, a near-optimal solution can be found in the form of an approximation algotirhm by relaxing the integrality constraint and rounding based on a fractional solution to the LP relaxation [193].

Table 5.2: Significant properties of the request-redirection scheme.

Properties	Parameters	Description	
Activation	Activation trigger (when)	Asynchronous (on CDN server request)	
	Activation decision (where)	Distributed (Gateway redirection upon requests from distributed servers)	
Implementation	Status information	Traffic load (correlated with server response load = utilization * capacity)	Alarm (Asynchronous feedback)
Redirection policy	Server selection (how)	Minimize redirection cost (mapping of overloaded and underloaded server lists)	
	Redirected entities (what)	End-user requests	

Table 5.2 summarizes the properties of the redirection scheme, according to the classifications presented by Cardellini et al. [42]. The first property specifies the mechanism in which redirection is activated and where the activation decision process is made. The next property focuses on the implementation, specifying the status information used for redirection. The last property delineates the redirection policy by stating the server selection strategy and the entities that are redirected. We describe these properties in more detail in later sections.

Dynamic load distribution algorithm. Figure 5.3 presents the pseudo-code for the proposed load distribution algorithm, named *LD_minCost*, which integrates the resource discovery, server selection, and request-redirection phases. It does not attempt to perform load distribution when the servers of a primary CDN are working under an acceptable load. At a given time, if an overload condition exceeding an alarm threshold is reached, as reported by a primary CDN server, servers' status (response load) is assessed and a list of lightly loaded servers is generated from each participating CDNs (Lines 1 to 5), making use of the resource discovery algorithm stated in the previous section. The traffic load and network proximity for each underloaded server are measured and the redirection cost R_c is calculated in Lines 6 to 10. The optimal server from the underloaded server list is selected such that R_c is minimized upon redirection and the server does not get overloaded due to steered traffic. The optimal server *opWS* is added to a set of usable server list *targetWSList*, maintained by the primary to satisfy its requests. Thus, a mapping is maintained between the requests and a set of suitable servers to serve those requests. As long as there is an

overloaded primary CDN server, a new server minimizing R_c (except the optimal server selected earlier) is added to this list (Lines 11 to 15). By using this strategy, we prevent a server from going into an overloaded state and multiple servers can serve a peak demand or a flash crowd situation. Moreover, if *targetWSList* contains several servers and their average load decreases significantly, one server is removed at a time from the list (Lines 17 to 21). It ensures that the degree of replication for serving requests does not remain unnecessarily high when requests relinquish over time. Moreover, it also guarantees that sufficient underloaded resources are always available in the CDN peering system so as to utilize them during load distribution.

```
     begin …
 1:  for all r ∈ R do
 2:    Obtain the list of available servers n, WSList from peers
       /* Resource discovery */
 3:    for i = 1 to n do
 4:      Populate o, OWSList and u, UWSlist
         /* Overloaded and underloaded server lists */
 5:    end for
 6:    for j = 1 to u do
 7:      Calculate incoming traffic load based on loadmetric
 8:      Measure network proximity
 9:      Calculate redirection cost Rc
10:    end for
11:      do
12:      Select optimal server opWS minimizing Rc
         /* Server selection */
13:      Add opWS to targetWSList
14:      Redirect request to opWS
15:     while alarm_flag = true
16:    if |targetWSList| > 1 then
17:      if targetWSList.avgLoad <= alarm_threshold/2 then
18:        Remove least loaded server in targetWSList
19:      end if
20:    end if
21:  end for
     ...
     end
```

Figure 5.3: Load distribution algorithm (*LD_minCost*).

5.3.3 Perceived Benefits

A major advantage of our approach over traditional DNS-based redirection systems is that the actual end-user requests (eyeballs) are being redirected, opposing to the local DNS requests as in DNS-based redirection. Therefore, we can achieve a finer grain redirection. Since load distribution is performed dynamically, any redirection change takes effect instantly. In contrast, due to the IP-address caching in the intermediate name server, the DNS dispatcher loses direct control on subsequent requests for a Time-To-Live (TTL) period following address resolution, and thus causes some delay before redirection changes have an effect [62]. We also seek to achieve high locality with good load balancing, since requests targeted to the primary CDN stay within its domain as much as possible and are redirected to least loaded peers only during peak load conditions. In this way, our solution does not produce widely oscillated outcomes due to load distribution through request-redirection, and thus we seek to achieve service "responsiveness". Finally, our approach is beneficial for larger Web objects in particular and it endeavors to neutralize any load imbalance in the system, since CDNs have dynamic nature to act in primary or peering roles. The performance results and simulation analysis in Section 5.5 support these claims.

5.3.4 Time Complexity

Let us consider a peering arrangement of one primary and P peers, with N users generating R requests in the system. Let n be the number of available resources, which are found during the lookup process. The time complexity for populating an available resources list from P peers is $O(P)$. Further, $O(n)$ is the worst case complexity for contacting n available resources by the primary to identify the servers that can serve the given content requests. Then the complexity for requesting service for one request from the available resources of the peers is $O(P+n)$. Therefore, the time complexity of the algorithm in Figure 5.1 is $O(PR+nR)$. A peer's server receives S service requests from the primary. A worst case scenario gives $S = R$, thus producing a time complexity of $O(nR)$ for all the n available servers to be used by the primary. Hence, the resultant complexity of the service request and response during resource discovery is also $O(PR+nR)$.

Let us consider that load distribution is performed among u number of optimally underloaded servers, from the list of n available servers. Given an overloaded condition remains for a constant time T, the complexity of the load distribution algorithm *LD_minCost* in Figure 5.3 is $O(nR+uR+T)$. By omitting the constant, the resultant complexity is given by $O(nR+uR)$. However, the resource discovery, server selection, and request-redirection are integrated to perform load distribution in the peering arrangement. Therefore, the overall time complexity is $O(PR+nR+uR)$.

5.4 Methodology

Measurement based performance studies may not reproduce the problems and scenarios for which the solutions are designed, since in real testbeds several important parameters, such as server and network load conditions, can not be controlled. It is also difficult to have a significant amount of geographically dispersed users simultaneously to generate traffic causing a flash crowd. An alternative can be conducting experiments on real nodes using a controlled network environment with a WAN emulator or Emulab network testbed. However, our main focus is on CDN peering, with the assumption that network is not the main bottleneck. In our context, simulation can provide a reasonably detailed representation of key system parameters. Therefore, we have developed a simulator, based on Independent Replication Method, using the CSIM/Java[3] simulation toolkit. It assists to conduct *repeatable* and *controlled* experiments that would otherwise be difficult to perform in real CDNs.

5.4.1 Simulation Environment and Parameters

In our simulation model is based on a reference scenario (Figure 5.4), which is an approximate representation of the environment, yet representative of the key system attributes. There is an established peering arrangement, consisting of four CDNs with their sets of Web servers placed at different geographical locations across the Internet. Each CDN has a set of servers and a pool of users to generate request streams. End-users request content via their browsers and make use of a proxy server according to the same client-side policy. In order to take part in peering, each CDN defines

[3] It creates process-oriented discrete-event simulation models. Please check: http://www.mesquite.com

a subCDN with a subset of its resources. To provide an accurate characterization of the scenario, we have simulated the main system entities: (i) Web servers, (ii) mediator, (iii) distributed SR, (iv) congestions, and (v) end-users. In our simulations, PA and PR have limited functionality for SLA-based negotiation, policy classifications and policy enforcement (Chapter 3). Since our goal is to measure the impact of request-redirection for distributing load in CDN peering, we refrain from simulating more detail of the network, i.e. Internet connections, network hierarchies, and narrow network bandwidth in the last mile, by using a network topology generator.

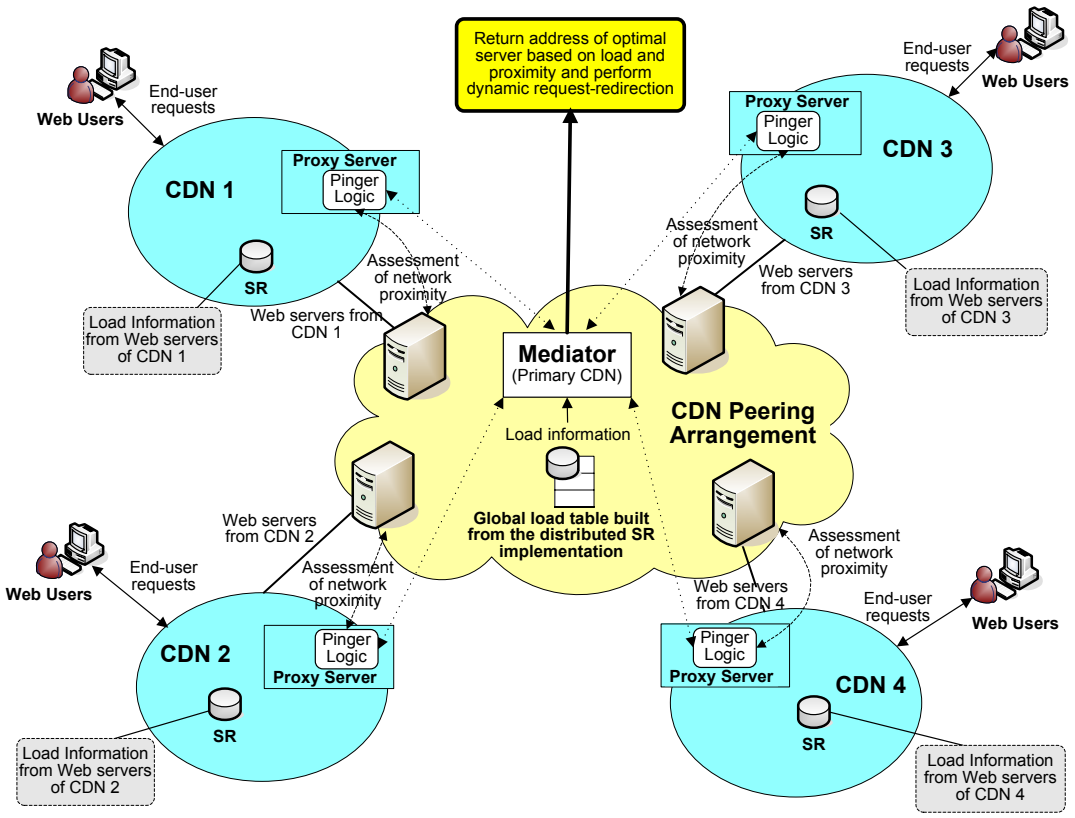


Figure 5.4: Reference Scenario for simulation.

Table 5.3 reports the system parameters used in our analysis. The parameter values, indicative of the simulation model, are chosen as follows. Each server calculates its utilization and updates the SR every 30s by sending an 11byes load index dissemination message. An alarm signal is sent if this value exceeds 80% of a server's utilization, i.e. alarm threshold. We use TCP for disseminating reliable and valid load index and UDP for network proximity measurement. The reason for using UDP in

the proximity measurement is because there are devices which will prioritize ICMP traffic (in case of TCP) over other traffic, which if there is congestion along the way, could skew the load. Some service setups such as firewalls and routers also limit ICMP traffic because of various denial of service threats. Since we intend to measure proximity under "realistic" traffic, using something closer is deemed significant. UDP also does not require acknowledgement of the received packets, causing less messaging overhead than TCP. We use an 18bytes proximity measurement query with a timeout period of 1000ms. To capture the unreliable nature of UDP, we model 30% packet loss rate. Finally, in order to reflect the effects of network congestions in our model, we use 100ms average network delay and 200ms average inter-CDN delay.

Table 5.3: Parameters for the system model.

Parameter	**Value**
SR update frequency	30s
Size of load index dissemination message	11bytes
Size of proximity measurement query	18bytes
Timeout period for proximity measurement	1000ms
UDP packet loss rate	0.3
Average network delay	100ms
Average inter-CDN delay	200ms
Alarm threshold	80% of a server's utilization
Traffic (end-user) distribution	$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \alpha, k > 0, x \geq k$

CDN Web servers. We have implemented the CDN servers as a set of facilities[4] that provide services to end-user requests. We have configured the servers according to the specifications from Fourth Quarter 2006 SPECweb2005 Results[5]. Table 5.4 summarizes the configurations of the Web servers. To anonymize this list, we have assigned a numeric identifier in the form WS_i, $i \in \{1,2,...,M\}$ to each server. We have used the domain part of the DNS name to distinguish between providers and assigned a provider identifier in the form CDN_j, $j \in \{1,2,...,N\}$ to illustrate which server is acquired from the same CDN provider. It has been found by Amini et al. [9] that a CDN provider may deploy unique IP addresses to its geographically diverse servers. Therefore, the distribution of the list of simulated Web servers follows such deploy-

[4] Each facility is a simulated resource with a single server and a queue for waiting requests.
[5] Standard Performance Evaluation Corporation. http://www.spec.org

ments. Column 4 of Table 5.4 presents the capacity of the Web servers, expressed as their ability to serve the maximum number of requests/second. To calculate the capacity for each server, we normalize the total number of multiple user sessions (SPECweb2005 reported performance) by the number of connected client machines as reported in the results. We model the servers to follow different service distributions with parameter values representative of the reference scenario. Column 5 specifies the used service distributions along with their PDFs and associated properties.

Table 5.4: Configuration of the simulated CDN Web servers.

Server ID	Provider ID	System Properties	Capacity (Req/s)	Service Distribution, PDF and Properties
WS_1	CDN_1	Dell PowerEdge 2950, Intel Xeon 5160 Processor	906	Pareto, $\alpha k^{\alpha} x^{-\alpha-1}$ $\alpha = 1.25$, $k = 1$
WS_2	CDN_1	Dell PowerEdge 2950, Intel Xeon X5355 Processor	1052	Pareto, $\alpha k^{\alpha} x^{-\alpha-1}$ $\alpha = 1.25$, $k = 1$
WS_3	CDN_1	Dell PowerEdge 860, Intel Xeon 3070 Processor	506	Pareto, $\alpha k^{\alpha} x^{-\alpha-1}$ $\alpha = 1.25$, $k = 1$
WS_4	CDN_2	Fujitsu PRIMERGY TX150 S5, Intel Xeon3060 Processor	200	Log-normal, $\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$ $\mu = 0.9681$, $\sigma^2 = 1.5846$
WS_5	CDN_2	Fujitsu PRIMERGY TX300 S3, Intel Xeon5355 Processor	310	Log-normal, $\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$ $\mu = 0.9681$, $\sigma^2 = 1.5846$
WS_6	CDN_3	HP ProLiant DL145 G2, AMD Opteron 285 Processor	471	Hyper-exponential, $\sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i x}$ $\mu = 0.5967$, $\sigma^2 = 2.6314$
WS_7	CDN_1	HP ProLiant DL380 G5, Intel Xeon 5160 Processor	552	Pareto, $\alpha k^{\alpha} x^{-\alpha-1}$ $\alpha = 1.25$, $k = 1$
WS_8	CDN_3	HP ProLiant DL585 G2, AMD Opteron 8212 Processor	624	Hyper-exponential, $\sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i x}$ $\mu = 1.65$, $\sigma^2 = 3.70$
WS_9	CDN_3	HP ProLiant DL585 G2, AMD Opteron 8220 Processor	843	Hyper-exponential, $\sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i x}$ $\mu = 1.10$, $\sigma^2 = 5.65$
WS_{10}	CDN_4	HP ProLiant DL585, AMD Opteron 885 Processor	660	Erlang, $\frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$ $\mu = 4.529$, $\sigma^2 = 0.321$

The response load of a Web server (*LoadMetric*) is expressed as a product of its utilization in [0, 1] at a given time during simulation and the maximum number of served

requests/second (capacity). We use an asynchronous feedback mechanism, which assists the Web servers to trivially measure their actual loads and periodically update them in the SR. If a server's load exceeds a given alarm threshold, it signals the mediator to perform load distribution. A normal signal is sent when the load returns below the threshold. The use of such an asynchronous feedback mechanism suffices to consider a server as a candidate for receiving requests only if that server has not declared itself critically loaded.

CDN servers (facilities) have no queuing delay, since they are configured to have high capacity and large bandwidth. This approach leads to the logical implication that servers in the simulation can handle any size of load. This assumption is necessary to deal with request arrivals with very large processing requirements [8]. The load distribution algorithm deployed in the mediator decides, upon receiving an alarm signal, whether a server is overloaded or not, depending on its load information from the global load table maintained by the distributed SR.

Distributed SR implementation. There is a distributed implementation of the SR in our simulation, wherein each CDN has an SR instance to get the load status of its Web servers. Once a peer delegates a set of its servers to define the subCDN, the real-time state of each surrogate is passed to a global load table of the SR using standard load metrics (*LoadMetric*). The global load table stores the overall state of the servers in the peering arrangement using a tuple (*Web Server*, *LoadMetric*).

Two key data structures are maintained to realize the distributed SR implementation. First, each SR instance in a peer is responsible for keeping a local copy of the load status of its servers. Load index is not necessarily propagated straightaway to the CDN peering system. Second, the primary CDN maintains a global load table to store the system load. Therefore, it globally communicates load index amongst the peers so that intelligent load distribution decisions may be made. The asynchronous feedback by the servers is used for consistency of global load information. Such best-effort data consistency mechanism allows proceeding with the operations without querying the servers for updated load index. Thus, the servers are made to decide on the tradeoff between serving requests and updating load information.

Maintaining a data structure for the global load table in the primary CDN may seem infeasible with regard to security and fault tolerance. However, assuming a set of trustworthy participants, mendacious behavior from a provider, posing security threats, is not expected to be usual. This assumption can be justified by the fact that a given peering arrangement is likely to contain a handful of providers. The overhead of load update messages could be high if a large number of Web servers from the participating CDNs are present in the system. Nevertheless, at an instant it is unlikely to have more than a few hundred or thousand nodes in a peering arrangement from a small number of participants. The other mitigating factor is that load information is updated periodically and the only data that needs to be sent is a few bytes of load index. For example: a total number of 1000 servers in a peering arrangement, 30s update frequency and 11bytes of data transfer for each load table entry "*WebServer_ID LoadMetric*" will lead to about 22KB/min, or < 0.5KBps for the messaging overhead due to the asynchronous feedback by the servers to the SR.

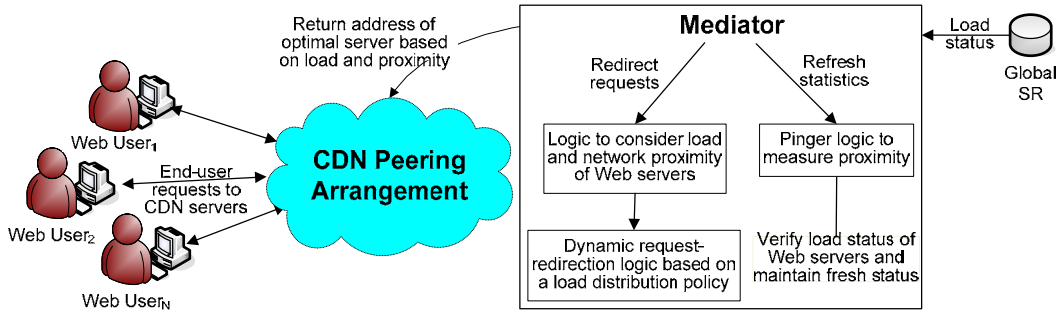


Figure 5.5: Operations of the mediator.

Mediator. Figure 5.5 illustrates the workings of the mediator, which is simulated to act as an authoritative entity in a given peering arrangement. It monitors the global load table to get Web servers' status and uses the pinger logic to get their network proximity information, in terms of round-trip response time (in milliseconds), assuming that they are directly correlated. The pinger logic uses the User Datagram Protocol to send an 18bytes proximity measurement query (as UDP packet) of the format "*PING Time CRLF*" to each Web server for the network proximity measurement. *Time* represents the timestamp when query is sent and *CRLF* repre-

sents the carriage return and line feed characters that terminate the query message. We also set a timeout period of 1000ms to check whether the servers are reachable. Thus, along with the proximity measurement, the pinger logic tests a Web server's ability to respond to a proximity measurement query, as well as its level of responsiveness under the current load.

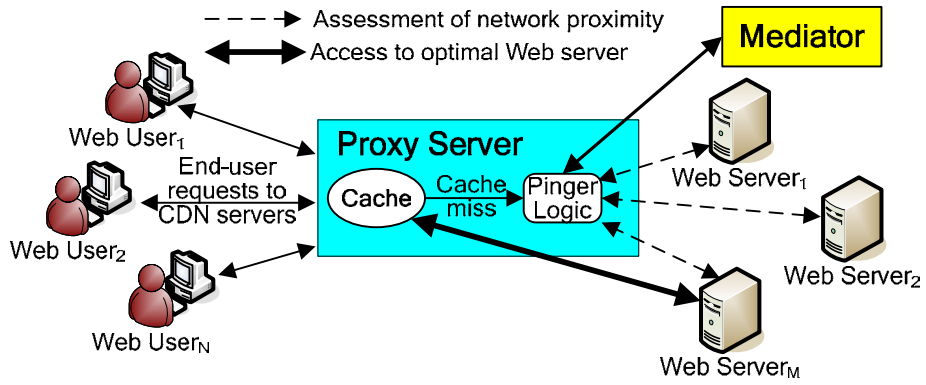


Figure 5.6: Network proximity measurement.

The evaluation of network proximity among end-users and edge servers is a function of network topology and dynamic link characteristics. Therefore, the pinger logic is deemed to be located close to the users so that the mediator ideally can have the same view of the network status as the user's browser (Figure 5.6). This approach allows estimating, reasonably accurately and possibly offline, the network proximity between the user and CDN Web servers. It may appear that this approach leads to a load distribution system that does not scale enough as it requires an instance of pinger logic to be placed close to each of the numerous number of users in the Internet. However, in practice it is observed that most of the end-users make use of a proxy server which filters Web accesses through caching. Content that is not available in the proxy server is retrieved from the origin server(s) and stored locally in a cache. Additional requests for the same content are served by the proxy until the expiration time after which the content is considered 'stale'. Therefore, the scalability problem can be solved by placing the pinger logic in the proxy server. This solution is feasible because the pinger logic is activated by the mediator for network proximity measurement only when the requested content is not in the proxy cache.

Network congestions. In order to consider the impact of network congestions on load distribution strategies, we model two types of networks delays, namely, *inter-CDN* and *intra-CDN* delays. In addition, for network proximity measurement we have simulated the UDP packet loss (*LOSS_RATE* = 0.3) due to network congestions. Intra-CDN delays have three components: (i) minimum round-trip time between server and the user browser, (ii) queuing delays at the mediator[6], and (iii) packet transmission time for each link on the user browser and CDN Web server path. While (i) and (ii) are measured through simulation, (iii) is modeled as average network delay of 100ms. Inter-CDN delays are random variables that model communication latency between different geographical CDN domains. The Inter-CDN delays are 200ms in average. Within the simulation model, we do not characterize the delays spent for address resolution of Web servers.

End-user traffic. User requests are implemented as CSIM processes[7]. We assume that users request content via their own browsers to the CDN, according to the same client-side policy. The hidden load weight [62] is implicitly taken into account through the user distribution, as requests to different CDNs are properly weighed and are distributed to the servers, according to a Zipf distribution with parameter α = 0.8, corresponding to a highly skewed function [61, 169].

Alike the Internet access workloads, end-user requests show self-similarity. A self-similar process has observable bursts in all time scales. It exhibits long-range dependence, where values at any instance are typically correlated with all future values. This self-similar nature can be described by using a heavy-tailed distribution [68, 69]. Therefore, we model the end-user requests to each CDN server to follow a highly variable Pareto distribution with the PDF,

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \alpha, k > 0, x \geq k$$

where the weight of the tail of the distribution is determined by $\alpha < 2$.

With this model, a realistic simulation environment for CDN peering is achieved.

---

[6] The queuing delay at the mediator occurs for any possible congestion during load distribution in the CDN peering system.

[7] CSIM processes are objects, based on Java threads, which make use of simulated resources.

Table 5.5: List of Performance indices.

Performance Index	Description
Concurrent flows	Number of ongoing requests at each server. A desirable scheme should always keep the number below each server's capacity limit.
Completions	Number of completed requests at each server. It is used to evaluate the performance of the proposed resource discovery mechanism.
Rejection rate	The percentage of dropped requests due to service unavailability. It assists in investigating the service disruptions in each scheme.
Utilization	A server's utilization in [0, 1] as reported by the load monitoring apparatus. It is used to demonstrate the performance of the proposed redirection scheme.
Maximum utilization	Highest utilization at a given instant among all primary CDN servers. It is used to emphasize the impact of redirection on load distribution of primary CDN servers.
Cumulative frequency of maximum utilization	The probability that the maximum utilization of the primary CDN is below a certain value. It is a major performance criterion which determines whether the primary CDN is overloaded or not, by focusing on the highest utilization among all primary CDN servers.

5.4.2 Schemes and Metrics for Comparison

Traditional use of DNS scheduler for load balancing generally applies the Round-Robin (RR) algorithm to map requests to servers [62]. Therefore, we use an RR-based and a probabilistic version of this policy to assess the effectiveness and to evaluate the performance of our approach. We also use a simple deterministic scheme for comparison purposes. Specifically, we experiment with the follow policies: *LD_RR*, *LD_PRR*, and *LD_LL*. When incoming load exceeds the alarm threshold, the LD_RR policy uses a Round-Robin approach to redirect excess requests to all available underloaded servers in a cyclic order. The LD_PRR policy is a variant of the LD_RR policy. The basic idea is to make probabilistic round-robin type assignment to the servers. The probability is based on the residual capacity of servers in a given peering arrangement using the latest server load index. For this purpose, we generate a random number v ($0 \leq v \leq 1$), and under the assumption that i–1 is the last chosen server, we assign the new requests to server i with *loadmetric* (utilization) u_i, only if $v \leq u_i$. Otherwise, we skip the server i and consider i+1 repeating the same process. The LD_LL policy uses a trivial approach that performs load distribution by redirecting

excess requests to the least loaded server. Just for comparison purpose, we also consider no redirection, which tries to assign requests to the closest server without considering the load. Table 5.5 lists the performance indices that are used in the experimental evaluation.

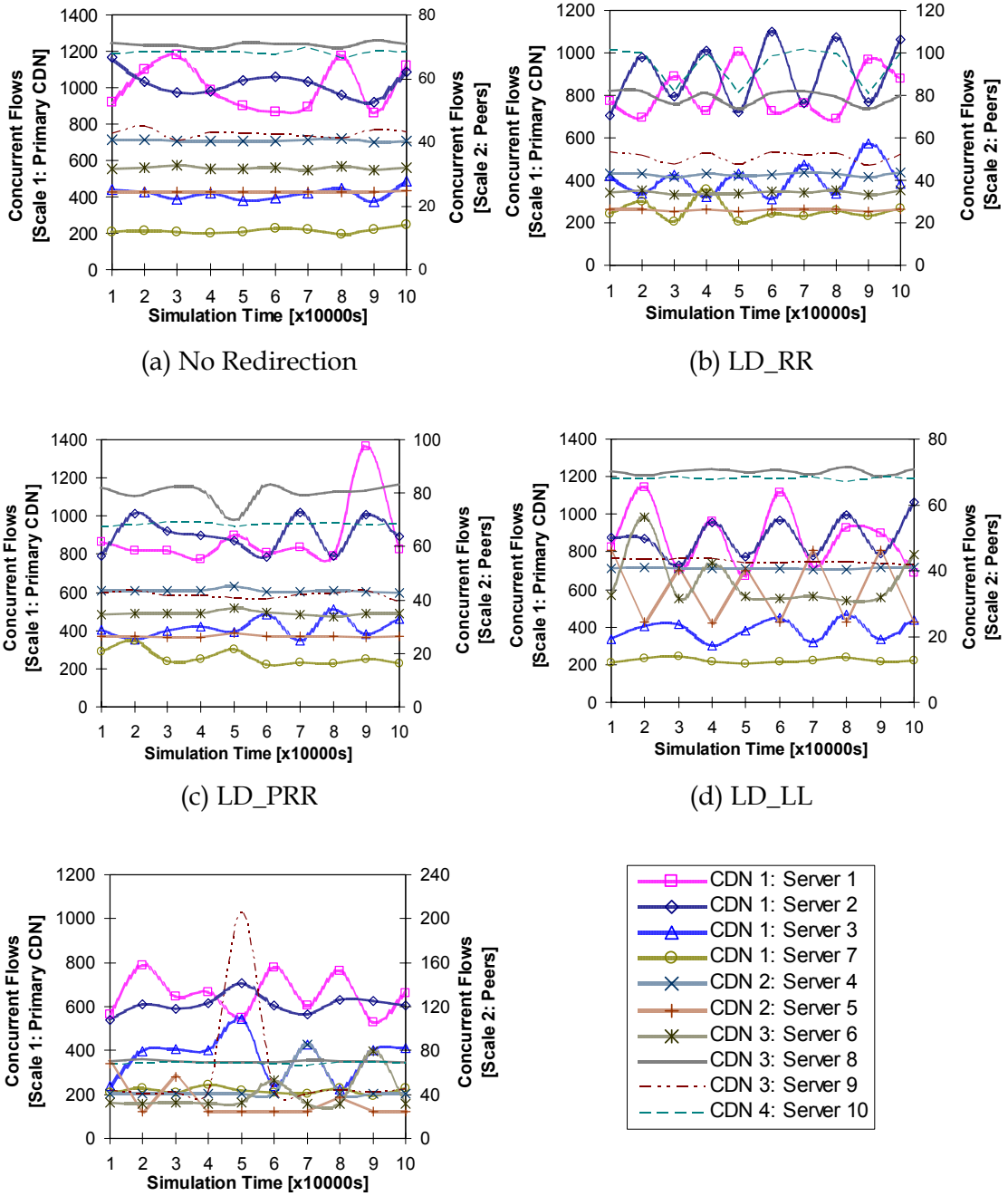
5.5 Experimental Evaluation

In this section, simulation results are presented to evaluate the performance and to provide critical assessment of our approach. We run our experiments for the reference simulation model of Section 5.4, with one provider as primary (CDN 1) and others as peers. Results are obtained from ten simulation runs, where the duration of each run is determined by the run length control algorithm built in CSIM. On average, each run counts for 10000s (approximately 3 hours) of the system activities.

While our simulations are designed to converge to the "true solution" of the model, running the experiments for a finite amount of time may not produce the exact true solution. However, choosing the right length of a simulation run is not obvious. Overly short simulation runs result in highly inaccurate performance statistics, whereas too long simulation runs unnecessarily waste computing resources and delay the completion of the simulation study. This problem can be address by estimating a confidence interval that is a range of values in which the true answer is believed to lie with a high probability. Therefore, we have calculated confidence intervals in order to show the accuracy in the results of simulation output. In our case, confidence interval with 95% confidence level is estimated to be within 4% of mean.

5.5.1 Traffic Load on Servers

Figure 5.7 shows the number of connections at each server. By keeping track of the concurrent flows at each server, we examine the performance of load distribution. These results indicate whether a scheme can keep the number of active connections (requests) to each server below respective capacity limit. For the clarity of presentation, we plot samples after each simulation run, using two scales—Scale 1 and Scale 2—to show the concurrent flows at servers of the primary and peers, respectively.



(a) No Redirection

(b) LD_RR

(c) LD_PRR

(d) LD_LL

(e) LD_minCost

Figure 5.7: Number of concurrent requests for each scheme.

Server load is not taken into account in the no redirection policy and user requests are sent to the closest server. Hence, from Figure 5.7(a) we observe that the load at a few servers, in the primary CDN domain, grows significantly. For example:

throughout the simulations, Server 1 and Server 2 receive more requests than their capacities. Unless they are provisioned with enough capacity to serve more concurrent connections, they will end up dropping many requests. In Figure 5.7(b), we see that LD_RR performs some load distribution by sending extra requests to the underloaded servers. However, it does not take cost (in terms of traffic load and proximity) into account and can potentially lead to high redirection cost.

Figure 5.7(c) presents the performance of LD_PRR. Since it assigns some probability to the underloaded servers based on their residual capacity, it redirects more requests to the server(s) with high probability. Therefore, it performs some load distribution but may cause sudden surge to a particular server, thus leading to imbalance load situations. For example: during simulation time 8 and 10 (x10000*s*), Server 1 receives many more requests than its capacity. LD_LL does not perform well as it fails to distribute loads to multiple servers. As for instance, from Figure 5.7(d), we observe that as simulation time passes, Server 1 receives more requests than its capacity, specifically, during simulation times 2, 4, 6, and 8 (x10000*s*), and Server 7 constantly receives extra requests than that it can handle.

In Figure 5.7(e), we present the performance of LD_minCost. The main objective of LD_minCost is to redirect extra requests in an imbalanced load situation, minimizing the redirection cost in terms of traffic load and network proximity, without violating the (practical) server capacity constraints. We observe that none of the primary CDN servers operate beyond their capacity during simulations. In order to prevent wide oscillation in the load distribution decisions, incoming requests to the primary CDN stay within its domain as much as possible and only cross the inter-CDN barrier during excessive load imbalance. Since redirection cost is taken into account, requests are served by the least loaded optimal servers, in terms of network cost. As a result, a few primary CDN servers receive only relatively few requests, while other better located servers run close to their capacity. In addition, request rejections are minimized as LD_minCost leads to optimal server selections. Moreover, any dynamic load changes, e.g. sudden load increase in Server 9 at simulation time 5 (x10000*s*), are also taken into account and load is distributed in a timely fashion.

Therefore, our approach performs well even under high traffic surges such as flash crowds.

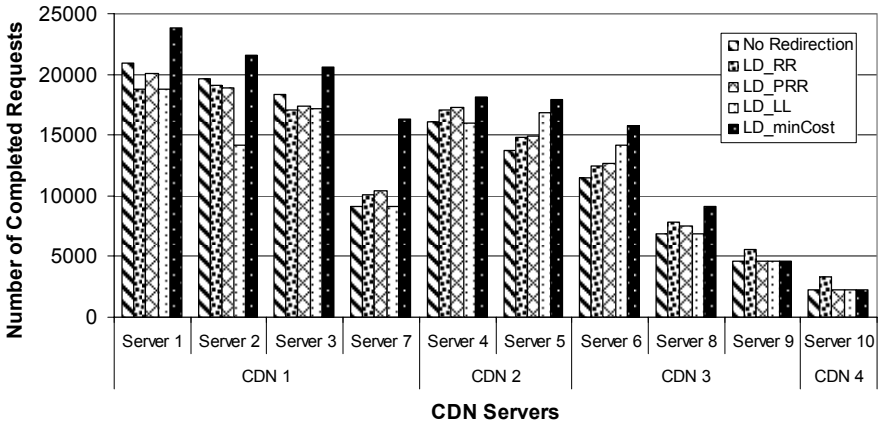


Figure 5.8: Number of completed requests at each server in different schemes.

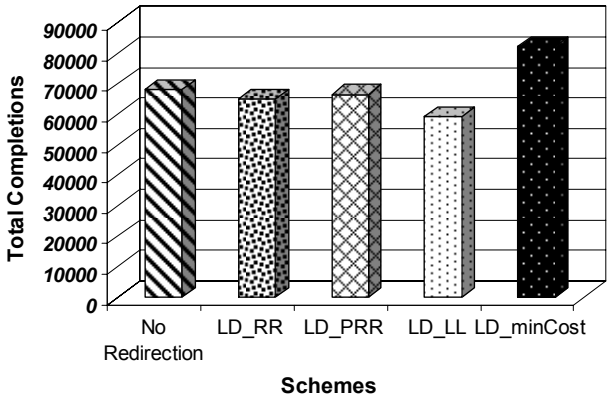


Figure 5.9: Total completions in each scheme.

5.5.2 Number of Completions

Now we evaluate the efficiency of our approach in terms of request completions. Figure 5.8 presents the average number of completed requests at each server over the simulation runs. It is found that in comparison to other alternatives, our approach is susceptible to handle more requests during load imbalance. LD_LL shows the worst performance among all the schemes. With no redirection, each CDN server attempts to serve the incoming requests to it, without any provision to redirect the request to a peer's server. Although servers are over-utilized during load imbalance, it does not perform well to serve all the incoming requests. Notably, LD_RR and LD_PRR show

almost similar performance in terms of the number of completions at each server. While it is expected that LD_RR and LD_PRR would exhibit better performance than no redirection, they perform as poorly as the no redirection policy. This is because many requests are dropped due to service disruptions. We further elaborate on this aspect with supporting results in the next section.

Figure 5.9 demonstrates a similar trend, which presents the total completions in the CDN peering system for different schemes. As adverted, LD_minCost assists in serving the highest number of requests collectively in the CDN peering system. LD_RR and LD_PRR, along with no redirection demonstrate almost similar performance in terms of the total completions. As expected, LD_LL performs the worst and leads to serving the least number of requests in the system.

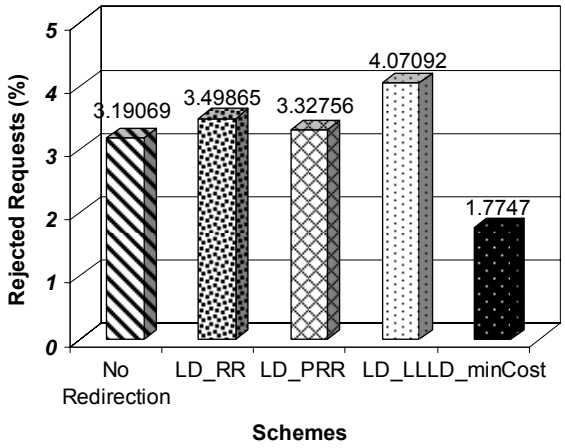


Figure 5.10: Service rejection rate for each scheme.

5.5.3 Service Disruptions

In this section, we investigate the impact of service unavailability for each scheme. Our redirection scheme directs comparatively more requests during load imbalance. On the contrary, LD_RR, LD_PRR, and LD_LL do not show a high redirection percentage under traffic surge. Eventually, many requests are dropped as incoming requests arrive to a primary CDN server and find that the server is operating at its highest capacity. Thus, the inability to redirect more requests due to limited server capacity leads to significant service disruptions in these schemes. It is evident from

Figure 5.10, which presents the percentage of disrupted services for each scheme, in terms of the average service rejection rate. In order to compute this performance metric, we first calculate the rejected service ratio as the number of requests that yielded a negative response (i.e. the system has not found a resource to serve this request), over the number of incoming requests. We then computed the average service rejection ratio as the average value over the number of total requests in the system. Figure 5.10 is obtained with a fixed number of 100,000 requests. From the figure, we observe that LD_minCost clearly outperforms other schemes, by exhibiting the lowest service rejection rate.

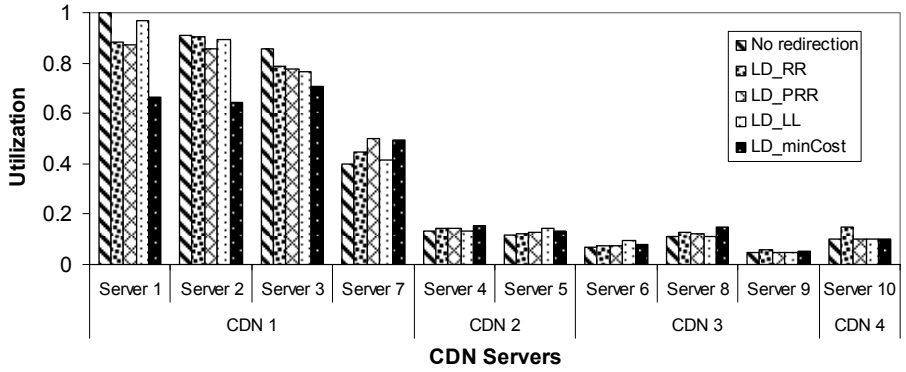


Figure 5.11: Server utilization for each scheme.

5.5.4 Server Utilization

We use utilization to determine whether the primary CDN servers operate under an acceptable level of load during high traffic surges. Figure 5.11 presents the utilization of the primary CDN servers for each request-redirection scheme. We observe that LD_minCost performs better than other redirection schemes for load distribution of the overloaded primary CDN servers. With no redirection, most of the primary CDN servers receive excessive traffic and utilization stays over the alarm threshold (80% of a server's capacity). While LD_RR and LD_PRR demonstrate similar characteristics to perform some load balancing, none of them reduces utilization of all the primary CDN servers below the threshold. LD_LL policy fails to perform load distribution among multiple servers and always overloads Server 1 and Server 2. On the other hand, LD_minCost exhibits the best performance by reducing the utilization of all the

primary CDN servers below the threshold. Since requests stay within the primary CDN domain as much as possible to minimize redirection cost, a better located server, e.g. Server 3 in Figure 5.11, may show more utilization than its allies. However, still the primary CDN operates under an acceptable level of load. It is also evident from Figure 5.12, which presents the average utilization of the primary CDN in each redirection scheme. LD_RR, LD_PRR, and LD_LL do not reduce the average utilization significantly below the threshold. Therefore, with these schemes primary CDN servers may be unable to receive more requests during sudden excessive traffic (flash crowds) in future and thus requests may have to be redirected outside the domain. With LD_minCost the average utilization of the primary CDN system is significantly brought down and makes its servers possible to cope up with succeeding traffic outbursts.

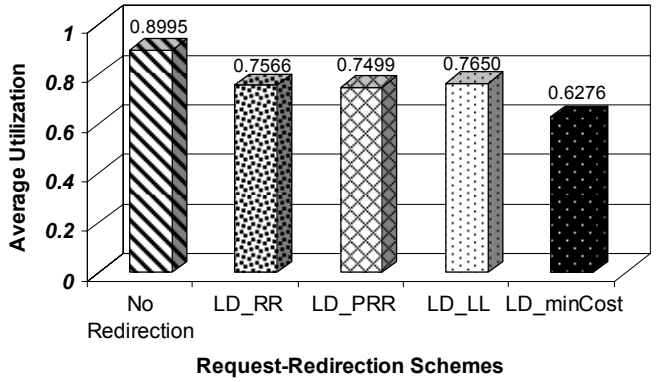


Figure 5.12: Average utilization of the primary for each scheme.

5.5.5 Redirection Performance

We investigate the impact of our request-redirection scheme on avoiding a primary CDN server that is overloaded. Hence, we do not adopt traditional metrics such as the standard deviation of server utilization for this purpose. We rather evaluate the performance of the request-redirection policies through the maximum utilization observed during a simulation run. The main performance metric we use is the cumulative frequency of maximum utilization, i.e. the probability for each utilization level that all server utilizations stay within that level. This metric provides an indication on the relative frequency of overloading. As for instance, a probability value of

0.8 for all servers to be less than 90% utilized implies that at least one server exceeds the 90% utilization level with probability 0.2.

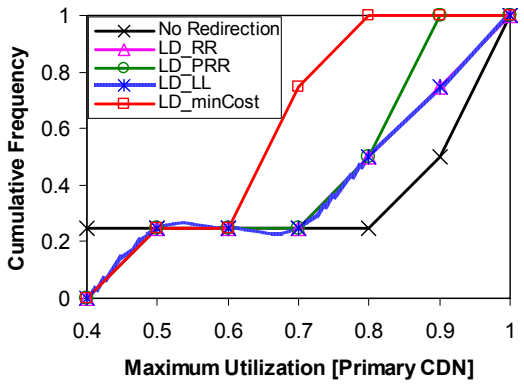


Figure 5.13: Comparison of the request-redirection schemes.

Figure 5.13 summarizes the performance of the request redirection schemes in terms of cumulative frequency of maximum utilization of the primary CDN servers. It shows that LD_minCost has a probability of 1.0 for not causing any primary CDN server to exceed 80% utilization (alarm threshold). From the figure we can see that other schemes such as LD_RR, LD_PRR and LD_LL do not perform well. Specifically, they exhibit a probability of 0.5 of not causing any Web server to exceed the threshold. Moreover, as predicted, with no redirection there is very low probability of only 0.25 that primary CDN servers will operate under the threshold.

5.5.6 Sensitivity Analysis

The performance of the redirection schemes can also be evaluated as a function of system parameters. We conduct a sensitivity analysis for the primary CDN, considering critical parameters such as *utilization* and *traffic distribution*. Changing other system parameters, such as the average number of requests or total simulation time, does not show noticeable differences among the redirection approaches. Therefore, we do not present those results here.

In Figure 5.14, we compare the sensitivity to the utilization for the primary CDN in each redirection scheme, using the probability that no server in its domain is overloaded as the performance metric. Therefore, we vary the alarm threshold from 0.75 to 1. We observe that our redirection scheme, LD_minCost, shows the best result in

all cases. Analogous conclusion can be drawn when we vary the distribution of users among the CDN servers, and obtain results based on 90% of the maximum utilization, i.e. the *Prob*(*Maximum Utilization* < 0.9). Figure 5.15 shows the probability that no server has a utilization higher than 90% as a function of the shape parameter α, which is varied from 1 (high variability) to 2 (moderate variability). Here, the performance metric (threshold) is changed from 80% to 90% to show the novelty of our scheme even under highly variable end-user request pattern.

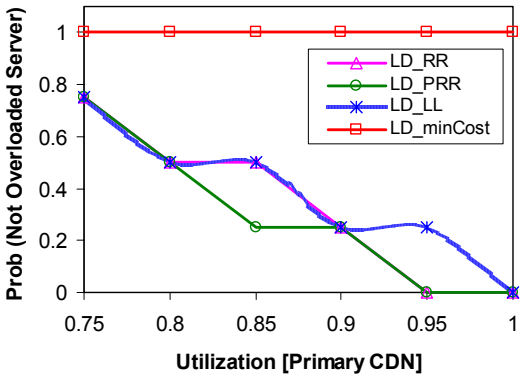


Figure 5.14: Sensitivity to utilization.

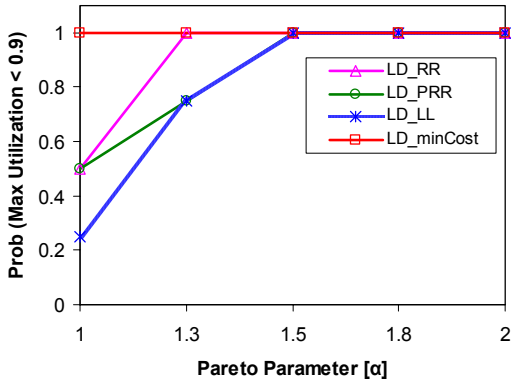


Figure 5.15: Sensitivity to traffic distribution.

5.6 Summary and Conclusion

In this chapter, we present resource discovery and request-redirection algorithms for a dynamic load distribution strategy, which alleviates any load imbalance in the CDN peering system. Resource discovery in our approach follows a distributed and

asynchronous nature, using a communication protocol that conservatively implements the publish/subscribe paradigm. In addition, request-redirection occurs over distributed sets of servers, minimizing redirection cost. Specifically, in our approach, when any Web server in a CDN peering arrangement reaches an overload condition exceeding the alarm threshold, the load distribution strategy reacts to redirect loads by selecting available least loaded server(s), while not compromising network proximity. We validate our proposal with the aid of simulations, considering practical constraints and significant system parameters; and demonstrate its performance using metrics such as the number of ongoing connections, number of completions, service disruptions, server utilization, and the cumulative frequency of maximum utilization. We also perform a sensitivity analysis of our redirection scheme by taking critical system parameters into account.

Our approach is novel as it seeks to perform load distribution in the CDN peering system through distributed resource discovery and dynamic request-redirection, taking network cost (in terms of load and network proximity) into account, and coping up with the practical constraints in the CDN domain. Our approach can alleviate problems with the commonly used DNS-dispatching policies for load balancing that do not provide sufficient control on end-user requests, e.g. it can have control on as little as 5% of requests in many instances [62]. Moreover, DNS-dispatching is less useful for fine-grain server selections. In contrast, we achieve a significant level of granularity, since the actual user requests (eyeballs) are redirected during load distribution among the servers in a CDN peering arrangement. Our approach endeavors to produce optimal server selections, even under high load skews.

Although there are many advantages, the proposed load and proximity-aware request-redirection scheme has some downsides. Since the actual user requests are redirected, it may incur additional round-trip latency. Specifically, fine-grain redirection for small objects may increase the latency. However, our approach is particularly beneficial for the delivery of large objects, e.g. software downloads, large on-demand files, and live streaming. It can be backed up by the evolving creation of more and more large Web content and media streaming integration in CDNs.

There is room for further improving our approach in terms of communication cost. While the asynchronous feedback mechanism has little communication overhead, it may suffer from unnecessary message passing at periods when the traffic load metric remain stable. As an alternative, we can use a triggered update strategy, where updates are triggered by the CDN servers as soon as they discover a potential change (measurement of old and new utilization and check whether it exceeds alarm threshold) in the end-user request patterns. Consequently, the communication overhead in the system is further reduced from periodic update, in order of the number of bytes/second.

Our approach could be aided with a market model that reflects the QoS-oriented aspects (user demand and satisfaction) for request servicing. This model can be used to analyze the sensitivity of performance metrics such as expected net utility [141, 204] of the involving entities with respect to various system parameters. Specifically, in the next chapter, we introduce a utility model for CDN peering and provide analysis on the measured content-serving ability of the system. This analysis provides incentives for the exploitation of different system parameters for a better CDN peering system design.

Chapter 6

Utility of CDN Peering

In addition to the ability to handle workload variations and thus allowing providers to expand their reach and capacity, recent trends foster the need for a utility model for the CDN peering system. Such a model endeavors to provide transparency, high availability, reduced investment cost, and improved content delivery performance. This chapter introduces a utility model and measures the content-serving ability of the CDN peering system. The proposed model provides a customer view of the system's health for different traffic types. It also captures the traffic activities in the system and helps to reveal the true propensities of participating CDNs to cooperate in peering. Through extensive simulations we unveil many interesting observations on how the utility of the CDN peering system is varied for different system parameters and provide incentives for their exploitation in the system design.

6.1 Introduction

The main value proposition for traditional CDN services has shifted over time. Initially, the focus was on improving end-user perceived experience by decreasing response time, especially when the customer Web site experiences unexpected traffic surges. Nowadays, CDN services are treated by content providers as a way to use a shared infrastructure with improved *utility* to handle their peak capacity requirements, thus allowing reduced investment cost in their own Web site infrastructure. Moreover, recent trends in CDNs indicate a large paradigm shift towards a utility computing model [39], which allows customers to exploit advanced content delivery services without having to build a dedicated infrastructure [92, 207]. Utility refers to

the quantification of a CDN's traffic activities and represents the usefulness of its replicas in terms of data circulation in its distributed network. It is vital as system wellness greatly affects the content delivery performance to end-users [163].

The CDN peering system is a type of content-utility system [207], characterized as offering efficient content delivery services with high availability, transparency, and improved performance without requiring content providers to build or manage complex infrastructure themselves. In this chapter, we exploit a utility-based *privileged provider* model to capture the content-serving ability of the CDN peering system via a utility measure. This is a monopolistic-natured model, where a CDN that initiates peering, i.e. primary, has the exclusive authority in the system. The measured utility can be translated into the usage benefits from the system. The simulation analysis of the proposed model reflects on the impact of system parameters on utility and provides insights for the CDN peering system design.

6.2 Motivation and Scope

Several recent trends demonstrate the emergence of the utility models for CDNs and general Web applications in the Internet. However, analysis of the prior work reveals only a modest progress in evaluating the utility for CDN peering. Prior work reflecting the utility computing notion mostly focus on the design and development of a utility computing platform. Specifically, Gayek et al. [92] provide an architectural framework for Content Serving Utility (CSU). Along with the component description, they provide performance results and key issues for designing and deploying the CSU. Subramanya et al. [207] propose a content-utility model for digital content delivery. They detail on the system overview and characterize its features. Canali et al. [39] present a Web-based utility computing model for Internet applications. They describe the approaches and challenges related to the design and development of a utility computing platform for CDNs. Our work is in line with these research efforts; however, we differ in that we not only provide an overview of the utility model for the CDN peering system, but also quantify the perceived utility.

There also exist research initiatives to perform simulation-based evaluation of

utility, as described in previous work [141, 204]. While the first explores a utility-based cumulative reputation system to encourage cooperation in a Peer-to-Peer (P2P) network, the latter set forth the usefulness of replicas and examine how single CDN utility is affected by various parameters. Our work is complementary to these work, as we provide simulation analysis of the proposed utility model for the CDN peering system. We differ by utilizing a privileged provider model to capture the content-serving ability of the system. We also reveal the impact of system parameters on the utility of CDN peering.

We foster the necessity and success of a well-designed content-utility system to provide highly scalable Web content delivery over the Internet. Utility of content delivery services can be measured using a representative metric that captures the traffic activities in a CDN, expressing the usefulness of its replica servers [204]. In the context of CDN peering, utility refers to the quantitative measure of the system-specific perceived benefit for content delivery. Customers interact with the CDN peering system in a limited number of ways and have little experience of the associated complex technologies. The responsibility of ensuring high performance content delivery is largely on the CDN peering system itself. Therefore, a comprehensive analysis of the impact of several system parameters on utility is important to improve the system's content-serving ability.

6.3 Utility Model for CDN Peering

The CDN peering system can be interpreted as a content-utility system [207], as shown in Figure 6.1. It comprises three main entities—*customers*, *service providers*, and *consumers*. Under this model, a provider can adjust resource provisioning for its content delivery services dynamically based on end-user demand.

Content providers are responsible for creating content, processing it to conform to certain formats, and developing content metadata. The CDN peering system as a service provider encapsulates several distinct CDN providers with storage and replication services to provide and manage content storage; network and communication services to enable swift and seamless transfer of content over com-

munications and data networks; and content delivery services to ensure easy and effective content consumption. Finally, consumers interact with the system by specifying the content/service requests through cell phone, smart phone/PDA, laptop, and desktop.

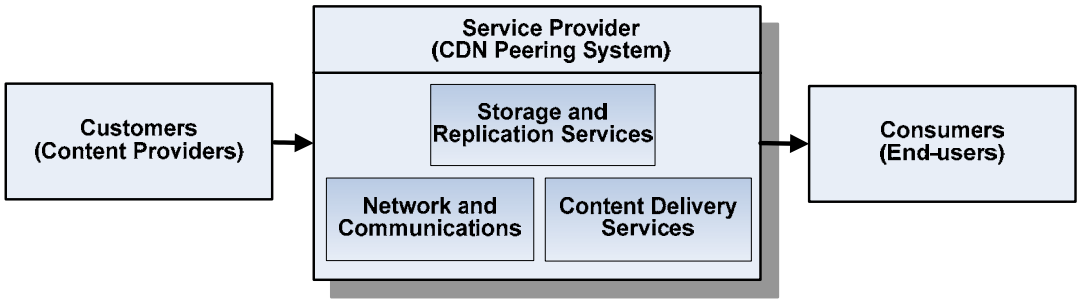


Figure 6.1: The CDN peering system as a content-utility system.

Figure 6.2 shows the major operational steps in the utility model for CDN peering. The system first captures/generates content from the customer (content provider). The generated content is then processed, indexed, and stored for online access. Upon receiving end-user requests for content, peers' resources are discovered using a communication protocol, reminiscent of the publish/subscribe paradigm (Chapter 5: Section 5.2.1). Under traffic surges, request-redirection is performed from the primary to the least loaded servers of peers. Consequently, the system then delivers the requested content to consumers in a swift and cost-effective manner. Once the system is fully in operation, its utility is disclosed to indicate to what extent it could fit an individual content provider's needs.

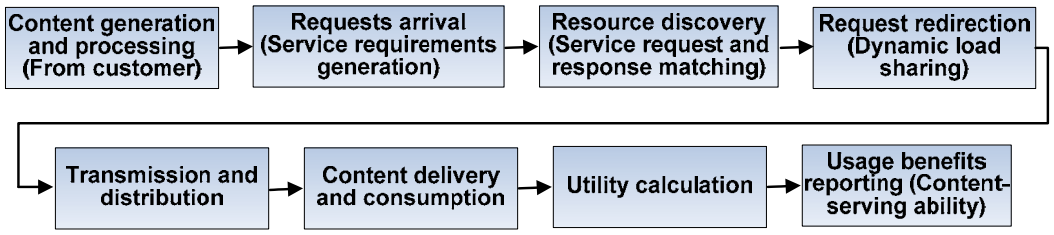


Figure 6.2: Major operational steps in the content-utility system.

An important benefit of this content-utility model is the availability of several services to enhance content usage and enrich end-users (consumers) experience with minimal burden. It also allows content providers (customers) to seamlessly interact

with the CDN peering system. The benefits for the CDN providers include the opportunity to exploit the availability of powerful, cost-effective services that offer customized content delivery.

6.3.1 Content-Serving Utility

To quantify the usage benefits of CDN peering, we develop metrics to measure its content-serving utility. Let us consider a CDN peering system comprising N replica servers from multiple participants, with the primary CDN having exclusive right to redirect requests for content. We use $R = \{r_j\}$, $j \in \{1, 2, \ldots, M\}$ to denote the set of user requests, with r_j being the j-th arriving request to the system. A utility metric u_{ij} expresses the value gained by a server i for serving an assigned request r_j according to the specified service requirements, i.e.

$$u_{ij} = \begin{cases} u_i & \text{iff } x_{ij} = 1 \\ 0 & \text{Otherwise} \end{cases}$$

where x_{ij} is the indicator variable to determine whether request r_j is assigned to server i according to the service requirements.

The most useful replicas for the CDN peering system are those exhibiting the highest utility. In this regard, we have drawn inspiration from Mortazavi and Kesidis [141], who use the notion of net utility to study the reputation of a node in a P2P system. We quantify the utility of a replica server with a value that expresses the relation between the number of serviced content requests against the number of rejected content requests. It is bounded to the range [0, 1] and provides an indication of the traffic activity. Formally, we quantify the utility u_i of a replica i to serve the assigned requests by using the following equation:

$$u_i = (2/\pi) \times \arctan(\zeta) \quad \textbf{(6.1)}$$

The main idea behind this metric is that a peer's replica is considered to be useful (high utility) if it serves content more than it rejects, and vice-versa. The parameter ζ is the ratio of the serviced requests to the rejected requests,

$$\zeta = \textit{No. of serviced requests} / \textit{No. of rejected requests} \quad \textbf{(6.2)}$$

The arctan function in (6.1) assists to obtain scaled resulting utility in the range [0, 1]. The value $u_i = 1$ is achieved if the replica does not reject any request. It can happen when the replica is working well under its capacity (i.e. almost idle) and ready to receive more content requests (yielding $\zeta = \infty$). The value $u_i = 0$ is achieved if the server is down and/or overloaded and cannot serve any request ($\zeta = 0$). In case of equal number of serviced and rejected requests, the resulting utility is 0.5. Ideally for the CDN peering system, the most cooperative replicas are those with high utility values.

By using individual server utility values and assuming that the requested content delivery services have been performed, we can compute the utility of the CDN peering system by taking the mean value of the yielded replica utilities. We refrain from using weighted average for this purpose as our aim is to reveal the true propensity of a CDN to cooperate in peering. For a CDN peering system governing N replicas from multiple providers, the content-serving utility U_P is:

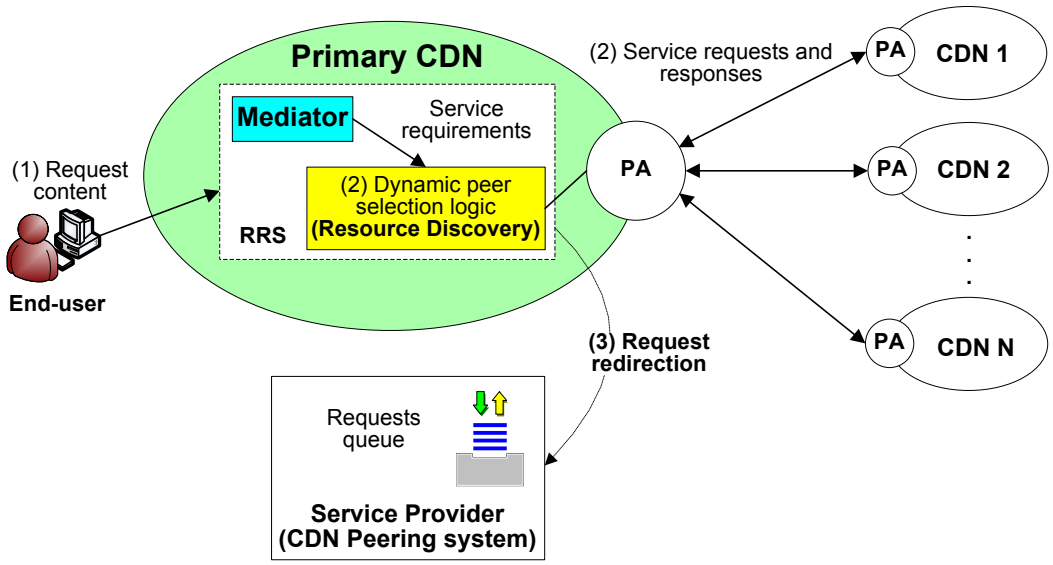
$$U_P = \sum_{i=1}^{N} u_i \Big/ N \quad \textbf{(6.3)}$$

The obtained utility using (6.3) can be translated into the content-serving ability (*durability*) of CDN peering. A highly durable CDN peering system exhibits high utility. This quantitative measure provides an indication of the effectiveness (health) and usage benefits of the system to content providers.

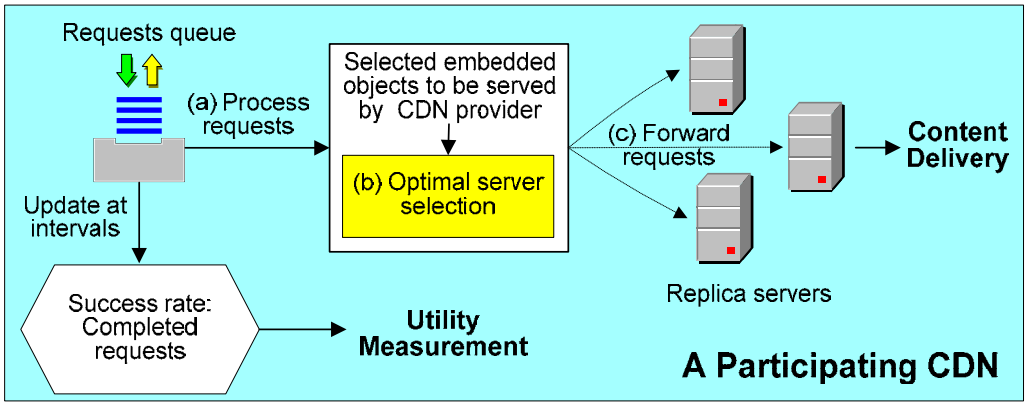
6.4 Evaluation Methodology

A schematic representation of the methodology used to evaluate the utility of CDN peering is provided in Figure 6.3. Herein we exploit the resource discovery and request-redirection mechanisms presented in Chapter 5. The simulation methodology realizes a privileged provider model (Figure 6.3(a)), with the primary CDN having the authoritative right over the resources it has acquired—which are delegated rights for the peers' physical resources. Content requests from end-users arrive to the Request Routing System (RRS) of the primary CDN. For certain content requests, under peak load or traffic surges (during a flash crowd event), users are redirected to the

least loaded server(s) of peers. Prior to redirection, matching of service requests and responses is performed to find target peers. Upon resource discovery, excess requests are offloaded to peers' server(s) in a cost-effective manner (in terms of traffic load and network proximity). To ensure scalability, redirection is performed in a per-flow manner, i.e. an optimal server is selected to accommodate multiple user flows. Requests for the same content from the same user group are aggregated and routed to the selected server(s). Any load imbalance in the system is alleviated through the repeated use of dynamic request-redirections [162] (Chapter 5).



(a) Simulation model



(b) Request servicing and utility measurement by a provider.

Figure 6.3: A schematic representation of the evaluation methodology.

The request servicing and utility measurement methodology in the CDN peering system by the selected optimal server of a participating provider is fleshed out in Figure 6.3(b). Our approach ensures that requests are serviced by the best responding server even under highly skewed load, while shifting traffic away from sites that are unreachable or near capacity. Simulations ensure that the system is autonomic and self-healing in the sense that if some sites are unusable, it moves traffic to others with no manual intervention. Each participating provider (peer) publishes aggregated individual success rate for satisfying incoming content requests at a time interval (epoch) during simulations so that replica utilities and consequent CDN utility can be measured. The CDN peering system in turn uses the individual utility functions to measure its utility and durability under variable incoming traffic. The outcomes can be conveyed to content providers as usage benefits of the CDN peering system.

6.4.1 Simulation Environment and Parameters

We resort to simulations to evaluate the utility model of CDN peering. We have used the simulator built using the CSIM/Java simulation toolkit, presented in Chapter 5, to conduct *repeatable* and *controlled* experiments.

Table 6.1: Parameters used for evaluation.

Parameter	**Value**
Number of CDNs	4
Sets of servers	10
Server capacity	Finite and heterogeneous
Service distribution among servers	Dissimilar. Pareto: $\alpha k^{\alpha} x^{-\alpha-1}$, Log-normal: $\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$, Hyper-exponential: $\sum_{i=1}^{n} P_i \lambda_i e^{-\lambda_i x}$, and Erlang: $\frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$
Traffic (end-user) distribution	$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \alpha, k > 0, x \geq k$
Traffic types	High to moderate variable, depending on the traffic distribution. Class 1 (a = 1), Class 2 (a = 1.2), Class 3 (a = 1.5), and Class 4 (a = 2)
Agreegated success rate refresh interval	1000s

As stated in Chapter 5, the simulation model is representative of a CDN peering environment and it is based on a reference scenario consisting of four CDNs with

their sets of servers placed at different geographical locations across the Internet. Each CDN has a set of servers and a pool of users to generate its own request stream. Users request content via their browsers and make use of a proxy server according to the same client-side policy. To take part in peering, each participating CDN defines a subCDN with a subset of its resources. To provide an accurate characterization of the scenario, we simulate the main system entities—Web servers, mediator, distributed Service Registry (SR), network congestions and end-user traffic. Table 2.1 reports the indicative system parameters for the simulation model.

CDN servers are implemented as a set of finite and heterogeneous capacity CSIM facilities, which serve incoming requests according to different service distributions. They are configured according to the specifications from Fourth Quarter 2006 SPECweb2005 Results (Chapter 5). We keep track of the number of active connections at the server side to calculate the aggregated success rate during simulations.

User requests are implemented as CSIM processes. Like the Internet access workloads, these user requests exhibit self-similarity. A self-similar process has observable bursts in all time scales. It exhibits long-range dependence, where values at any instant are typically correlated with all future values. This self-similar nature in user requests to each CDN Web server is captured by using a heavy-tailed [68, 69] Pareto distribution with the following Probability Density Function (PDF):

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \alpha, k > 0, x \geq k$$

where α determines the weight of the tail of the distribution.

6.4.2 Performance Metrics

We evaluate the utility of the CDN peering system under four types of request traffic—Class 1 (α = 1), Class 2 (α = 1.2), Class 3 (α = 1.5), and Class 4 (α = 2)—varying the request arrival from high (α = 1) to moderate (α = 2) variability. The reason behind using different traffic classes is due to the observation that subscribing content providers can be highly heterogeneous in terms of their traffic patterns and the type of content they handle [75]. Hence, end-users request for content of varying sizes (ranging from small to large). The processing requirements also vary based on size of

the requested content. The use of different traffic types allows us to reflect different user preferences and content request types. Consequently, these traffic types determine the behavior of processing in a given CDN's service capacity and influences the measured utility.

The primary focus of the study in this chapter is to provide observations on how the CDN peering system's content-serving ability is varied for different system parameters. The *utility* of CDN peering is measured according to the model in Section 6.3.1. Using the notion of utility, we express the traffic activity of the system. High utility value not only indicates the proficient content-serving ability of the system, but also signifies its durability even under highly variable traffic activities.

Table 6.2: List of performance indices.

Performance Index	**Description**
Utility	Content-serving ability, ranges in [0, 1]
Minimum utility	Lowest utility at a given instant in the CDN peering system
Cumulative frequency of minimum utility	The probability that the minimum utility of the CDN peering system is below a certain value
Response time	The time experienced by an end-user to get serviced
Completions	Number of completed requests
Rejection rate	The percentage of dropped requests due to service unavailability

To emphasize the impact of different traffic types on the measured utility, we report the *cumulative frequency* of the *minimum utility* (at a given instant) in the CDN peering system. We present for each level of utility the probability (or fraction of time) that the system realizes a given minimum utility for a certain traffic class. This metric provides an indication of the relative frequency of satisfying user requests for different traffic types. For example, if the probability of achieving 0.8 utility is 0.75, it implies that there is a probability 0.25 that the system realizes utility below 0.8.

We also measure the mean *response time* of each CDN, which specifies the average serving time of end-user requests. Lower values indicate fast serviced content. In addition, we keep track of the number of *completions* to show how a CDN is susceptible to different traffic types. Finally, we use the *rejection rate*, which depends on the number of disruptions due to service unavailability. Table 6.2 summarizes the performance indices that are used in the experimental evaluation.

6.5 Experiment Results

We run our experiments according to the methodology (Section 6.4) for a reference scenario (Chapter 5: Section 5.4.1), with one provider as primary (CDN 1) and others as peers. Results are averaged over ten simulation runs, where the duration of each simulation run is determined by the run length control algorithm built in CSIM. This approach endeavors to converge to the *true solution* of the simulation model in a finite simulation run. We avoid the adverse effects of both overly short and too long simulation runs, which may respectively cause inaccurate performance statistics, and unnecessary wastage of computing resources and delays in the completion of the simulation study. It is found that each simulation run is for approximately 3 hours of the CDN peering system activities. During a simulation run there are epochs of 1000s, at which the aggregated success rate of serviced requests is published. For all simulation results, confidence intervals[8] are estimated, and the 95% confidence interval is observed to be within 3% of the mean.

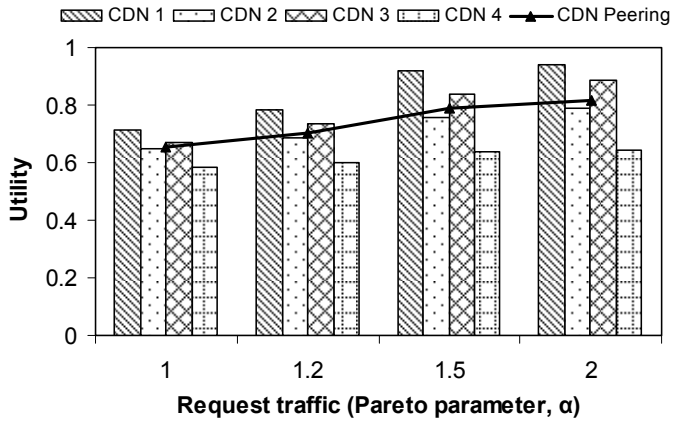


Figure 6.4: Utility measures for different traffic types.

6.5.1 Request Traffic vs. Utility

We first present the utility of the participating CDNs for different traffic types (Figure 6.4). For a provider, it is a normalized ratio (in [0, 1]) expressing the number of serviced requests against the number of rejected requests (Section 6.3.1). Each bar

[8] A range of values in which the true answer is believed to lie with a high probability.

represents different CDNs, while the bold line represents the utility of the CDN peering system as a whole. In general, we observe that altering from high to moderate variable request traffic results in lower to higher utility for the participating providers. We revisit this point in more details in the next sections.

In the CDN peering system, a CDN provider's utility implicitly depends on the number of its allocated servers and their associated service capacities. Moreover, optimal server selection using request-redirection [165], taking into account network proximity, congestion and traffic load of a server, also impacts the resulting utility in a CDN. It should also be noted that the use of different redirection policies by the primary to direct requests to the participating peers' server under different scenario may result in different utility values. Figure 6.4 shows that CDN 1 (primary) and CDN 3 (a peer) demonstrate higher utility than that of other peers. They contribute more servers (with higher capacity) to the system than other peers. In addition, server selection results in more requests to be redirected from CDN 1 to the servers of CDN 3, identifying it as a peer with close proximity to the primary.

While CDN 2 does not allocate as many servers as CDN 1 and CDN 3, it still exhibits higher utility than CDN 4. The reason behind this result lies in the difference of service distribution and capacity of CDN 4. Moreover, this peer contributes the fewest servers to the system. Optimal server selection does not produce as many target servers from CDN 4 as in other peers. Hence, it can only gain low utility for the fewer (in comparison to other peers) redirected requests. This leads to the logical implication that the contributing server of CDN 4 is distant in terms of network proximity, there might be congestion in the network path, and/or it has been suffering high traffic load. Our reasoning is justified in Section 6.5.3 with supporting results.

From Figure 6.4, it can be seen that a low utility value of CDN 4 significantly contributes to the overall utility of the CDN peering system. Since the resulting utility of the system is averaged over the individual utilities of participants, without the contribution from CDN 4 the system yields more utility value. Therefore, the primary may decide to either re-negotiate peering or exclude CDN 4 from peering in order to harness better content-serving ability from the system.

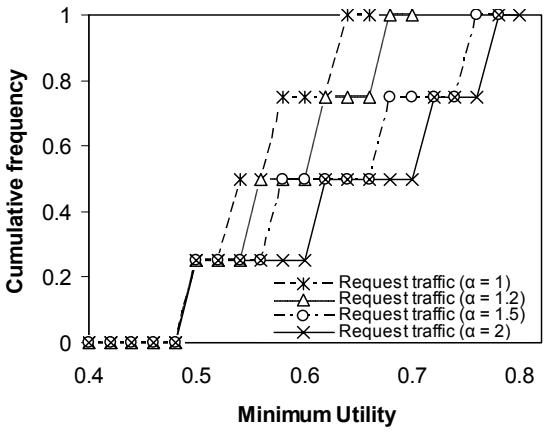


Figure 6.5: Cumulative frequency of minimum utility.

6.5.2 Content-Serving Ability

In this section, we focus on the content-serving ability of CDN peering. For that we investigate into detail the impact of different traffic types for the resulting utility in the CDN peering system. Figure 6.5 summarizes the effect of request distributions of candidate traffic classes on the achieved utility. The main goal is to show to what extent the system can satisfy user requests. For this reason, rather than adopting traditional metrics such as the standard deviation of utilities, we evaluate the primary CDNs performance under different traffic types through the minimum utility observed during simulation epochs. Figure 6.5 shows the cumulative frequency of the minimum utility as a major performance criterion. It indicates the probability (or fraction of time) that the system is able to achieve a given utility level. From the figure, it is visible that for moderate traffic (α = 2), CDN peering has a probability of 1.0 that it can realize little higher than 0.8 utility. The system is susceptible to traffic variability and thus exhibits lower utility values for heavy traffic. When the traffic distribution is highly variable (α = 1), the finite capacity servers of the participating providers fail to serve many requests. Specifically, under this traffic class, the CDN peering system can only achieve less than 0.7 utility. It establishes the reasoning that the utility of the CDN peering system is heavily dependent on the incoming traffic.

To report the individual content-serving ability of participants, we use the probability that the minimum utility is above 0.65, i.e. Prob (Min Utility > 0.65). Figure 6.6

presents the sensitivity to traffic distribution for each participating CDNs. We observe that the primary (CDN 1) realizes invariant performance for any traffic type (moderate to highly variable). It exhibits the maximum utility at all times and does not go below the minimum utility level. On the contrary, its peers (CDN 2 and CDN 3) are prone to the variability of incoming traffic. Specifically, they demonstrate diminishing performance with heavy traffic. Among all the peers, CDN 4 has the worst performance and shows close to 0.4 probability of gaining utility above the minimum utility level under heavy traffic demand with high variability. This is due to the fact that this peer drops many requests due to service disruptions. We further elaborate on the service disruption aspect with supporting results in Section 6.5.4.

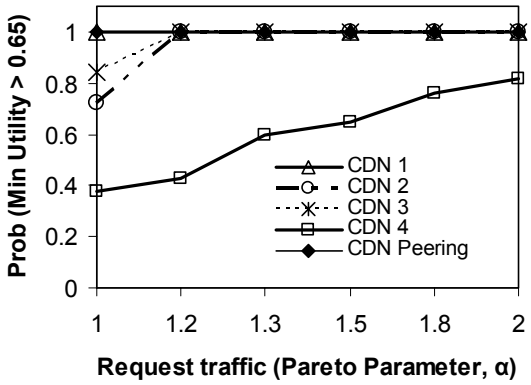


Figure 6.6: Sensitivity to traffic distribution.

6.5.3 Response Time vs. Utility

Figure 6.7 records the utility and mean response time of the participating CDNs. Two scales—Scale 1 and Scale 2—are used to respectively plot the response time and utility of each provider against different traffic types. The mean response time measure indicates the responsiveness of an individual CDN and the user perceived experience when accessing its servers.

From the figure it can be noted that response time and utility are inversely related. For a low response time, a high utility value could be achieved. For all the participating providers in the CDN peering system, we observe a similar trend. When a CDN is more responsive to incoming requests, end-users comprehend low response time, as fewer requests are dropped due to service unavailability. Although it is de-

sirable to achieve low response time and high utility at all times, variability in the incoming traffic impacts the response time and thus leads to higher response time and lower resultant utility. In particular, Figure 6.7 shows that the mean response time of CDN 2 and CDN 4 are more susceptible to incoming request variability. For highly variable incoming traffic (under traffic surges), the end-user perceived response times from these providers are increased, thus showing the evidence of likely network perturbations and heavy traffic load on their servers.

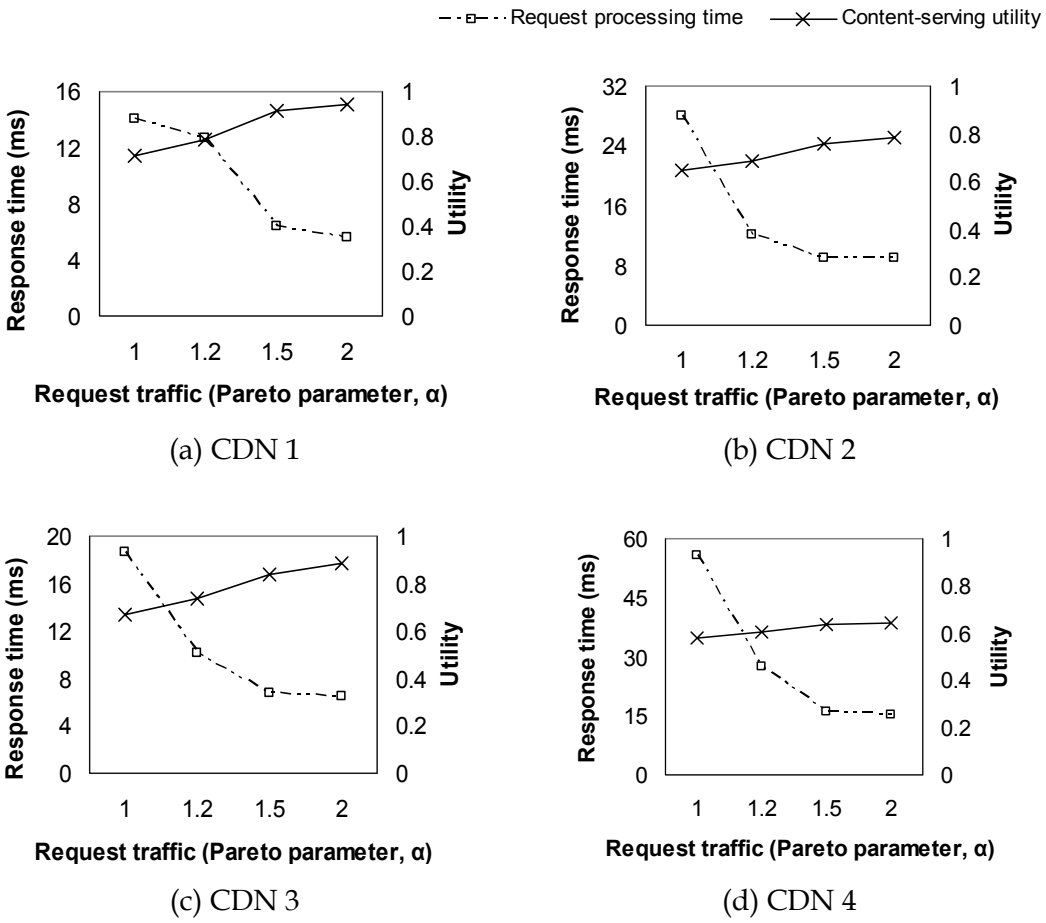


Figure 6.7: Relation between mean response time and content serving utility.

6.5.4 Service Completions

Now we investigate how the number of satisfied requests dictates the utility of CDN peering. For that we first study the number of request completions and rejections due to service unavailability. In our simulation, each server processes requests according

to its finite capacity. Figure 6.8 presents the average number of completed requests at each server over the simulation runs. It is found that with moderate incoming traffic, CDN servers attempt to serve more requests than that of the presence of highly variable request traffic. A similar trend can be observed from Figure 6.9, which shows the total completions in each participating CDN and in the CDN peering system.

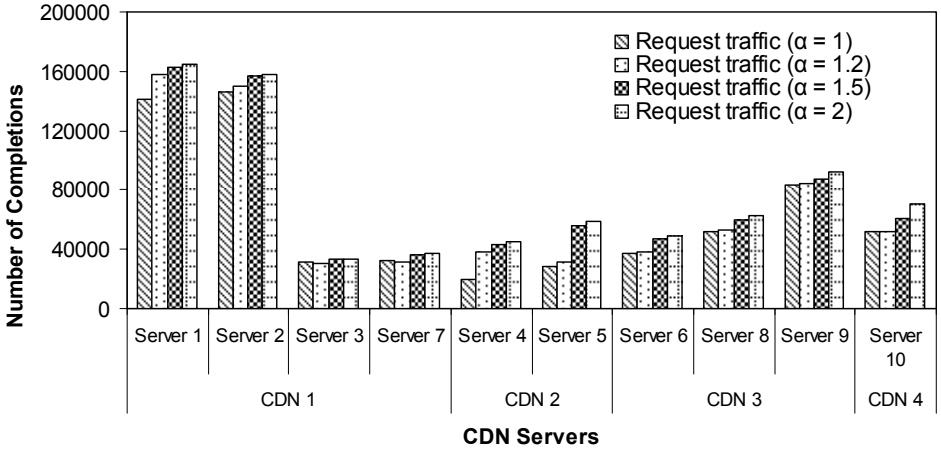


Figure 6.8: Number of completions in each CDN server.

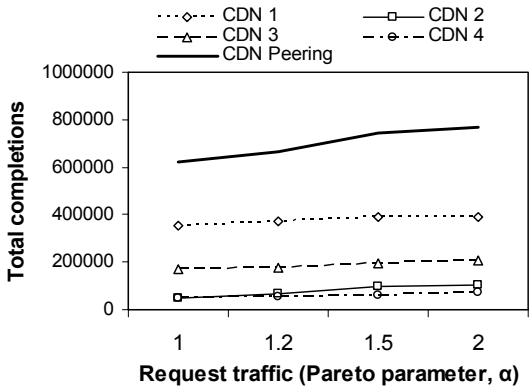


Figure 6.9: Total completions in each CDN and in the CDN peering system.

If a server receives more requests than its capacity, unless it is provisioned with enough capacity to serve more concurrent connections, it will end up dropping many requests. Since the servers are configured to operate below their finite capacity, they suffer from service disruptions under traffic surges with highly variable incoming requests. Consequently, many requests can not be served as incoming requests arrive to a CDN server and find that it is operating at its highest capacity. Figure 6.10 pre-

sents the average percentage of disrupted services for different traffic types at each epoch during the simulation runs. This figure is obtained with a fixed number of 1,000,000 requests. Service disruptions are expressed in terms of the average service rejection rate. To compute this performance metric, we first calculate the rejected service ratio as the number of requests that yielded a negative response (i.e. the system has not found a resource to serve this request), over the number of incoming requests. We then compute the average service rejection ratio as the average value over the number of total requests in the system. From Figure 6.10, it is evident that finite server capacities lead to service disruptions, which is more significant for highly variable traffic. We observe higher service rejection rate than that reported in Chapter 5. This difference is realized as we generate more requests than previously, with stringent requirements to publish aggregated success rate at regular intervals.

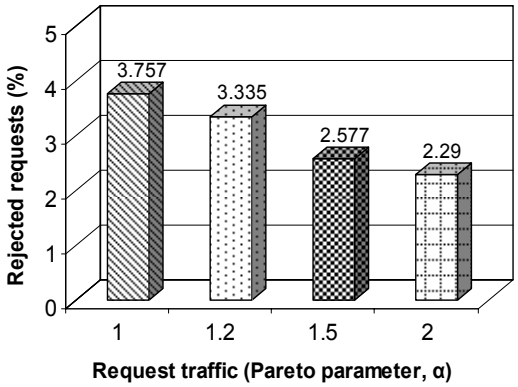


Figure 6.10: Service rejection rate in the CDN peering system.

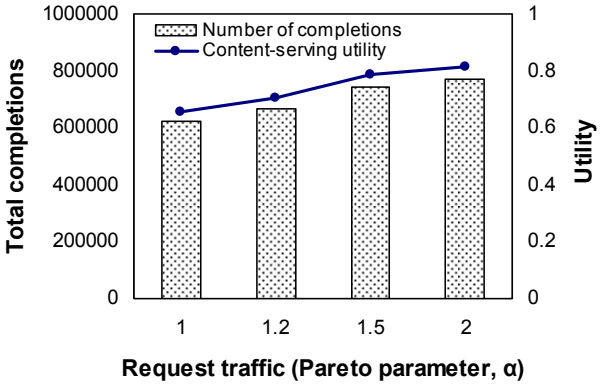


Figure 6.11: Total completions and utility for CDN peering.

Finally, we demonstrate the impact of completions on the utility of the CDN

peering system. As conveyed earlier, the total number of completions and resulting utility lessens with highly variable traffic types. The observed trend is sensible as the number of serviced requests (completions) and associated rejections act as major parameters in our utility model. The supporting results are presented in Figure 6.11.

6.6 Summary and Conclusion

CDN peering is a content-utility system that improves site performance and availability without requiring content providers to build or manage complex content delivery infrastructure themselves. In this chapter, we introduce a utility model to measure the availability level and health (content-serving ability) of the CDN peering system under variable traffic. This measure is crucial as the system wellness greatly affects the delivery and consumption of content. The outcomes can be interpreted as the benefits for a content provider to use CDN peering. We also show that our utility model can assist to reveal the true propensity of a provider to cooperate in peering.

With the aid of simulation experiments, we analyze the impact of system parameters on the perceived utility. We show that although the CDN peering system observes high utility in terms of satisfying content requests, its content-serving ability is largely dependent on the participating providers utilities. These utilities are essentially tailored to individual provider's service capacity, proximity, network conditions, and incoming request traffic. Our observations can be exploited for a better system design to cope with high traffic phenomena such as the flash crowd events.

The experiment results are quite encouraging to spawn further study on utility for content delivery services in a practical system. In the next chapter, we demonstrate our work in this context. Specifically, we implement the proposed utility model in the MetaCDN [29, 157] system that makes use of the brokering-based model for peering (Chapter 3: Section 3.4.1). We also devise a utility-based request-redirection policy and define a pricing policy to empirically measure the content provider and system surplus.

Chapter 7

Optimizing Utility of a Content Delivery Cloud

In this chapter, we report the results of a *proof-of-concept* testbed experiment on MetaCDN [29] that realizes a Content Delivery Cloud [60, 154] by utilizing Cloud Computing [35] to provide content delivery services to Internet end-users. While it ensures satisfactory end-user perceived performance, it also aims to improve the traffic activities in its world-wide distributed network and uplift the usefulness of its replicas. As an enhancement to this system, we measure the utility of content delivery via MetaCDN, capturing the system-specific perceived benefits [157]. We use a novel utility measure to devise a request-redirection policy that ensures high performance content delivery. We also quantify a content provider's benefits from using MetaCDN. Empirical performance results are presented to reveal our observations on the MetaCDN utility and content providers benefits from using MetaCDN.

7.1 Introduction

The CDN industry, i.e. content delivery, consumption and monetization, has been undergoing rapid changes. The multi-dimensional surge in content delivery from end-users has lead to an explosion of new content, formats as well as an exponential increase in the size and complexity of the digital content supply chain. These changes have been accelerated by the economic downturn in the sense that the content providers are under increasing pressure to reduce costs while increasing revenue.

From the previous chapters we know that with the traditional model of content

delivery, a content provider is locked-in for a particular period of time under specific Service Level Agreements (SLAs) with a high monthly/yearly fees and excess data charges [99]. Thus, far from democratizing content delivery, most CDN services are often priced out of reach for all but large enterprise customers [186]. On the other hand, a commercial CDN provider realizes high operational cost and even monetary penalization if it fails to meet the SLA-bound commitments to provide high quality service to end-users. Thus, it suffers from—spiraling ownership costs; resource wastage for maintaining infrastructure; inability to grow or profit from economics of scale; inability to fully monetize new or long tail content—to leave lucrative business deals on the table and forfeit profits. Moreover, as discussed in Chapter 6, there is a trend shift in the main value proposition for CDN services, from uplifting end-user experience to making use of a shared infrastructure in order to reduce investment cost for content providers.

To break through these barriers, one approach is to exploit the recent emergence of "Cloud Computing" [35], which moves computing and data away from desktop and portable PCs into computational resources such as large Data Centers ("Computing") and make them accessible as scalable, on-demand services over a network (the "Cloud"). The main technical underpinnings of Cloud Computing infrastructure and services include virtualization, service-orientation, elasticity, multi-tenancy, power efficiency, and economics of scale. The client perceived advantages for cloud-services include the ability to add more capacity at peak demand, reduced cost, experiment with new services, and to remove unneeded capacity.

By harnessing resources from multiple storage cloud providers, the MetaCDN overlay system creates a Content Delivery Cloud [60, 157]. It provides an efficient content delivery solution by forming a high performance, reliable and redundant geographically distributed CDN.

7.2 Motivation and Scope

Extending the traditional CDN model to use clouds for content delivery, i.e. a Content Delivery Cloud, is highly appealing as storage cloud providers, e.g. Amazon

Simple Storage Service (S3), Amazon CloudFront, Mosso Cloud Files, and Nirvanix Storage Delivery Network (SDN), charge customers for their utilization of storage and transfer of content (*pay-as-you-go*), typically in order of cents per gigabyte. They also offer SLA-backed performance and uptime guarantees for their services. Moreover, they can rapidly and cheaply scale-out during flash crowds [14] and anticipated increases in demand. However, unlike a fully-featured CDN, not all the storage cloud providers offer capabilities for intelligent replica placement, automatic replication, fail-over, geographical load redirection and load balancing.

MetaCDN provides the required features for high performance content delivery via an on-demand cloud service, eliminating costly capital expenditures or infrastructure upgrades. MetaCDN can be deployed as a fully outsourced, end-to-end services platform or as a complement to a CDN provider's existing infrastructure. Thus, it provides flexibility to CDN providers and their customers (content providers) to tailor a solution to meet their unique needs.

A vital component for MetaCDN is a request-redirection technique for directing end-user requests to optimal replica servers according to performance requirements. A suitable redirection mechanism extends the system's reach and scale and can alleviate the problems with overloaded servers and congested networks to maintain high accessibility [19]. Therefore, it is desired to devise a request-redirection mechanism that exhibit the following properties—scalability, transparency, geographic load sharing, cost-effectiveness, and high user perceived performance, to name a few.

MetaCDN utilizes intelligent request-redirection to choose an optimal replica for content delivery, thereby ensuring satisfactory end-user perceived performance. Given the responsibility to ensure high performance advanced content delivery services, a crucial goal for MetaCDN should be to improve the quantitative measure of *utility* for its services. While the notion of utility is enticing in relation to content delivery, from Chapter 6, we have seen that only a few previous work [39, 92, 141, 204, 207] have considered it. Thus, we focus on developing a redirection policy for MetaCDN based on the measured utility to ensure high content delivery performance, and implementing it in the existing MetaCDN system as a proof-of-concept.

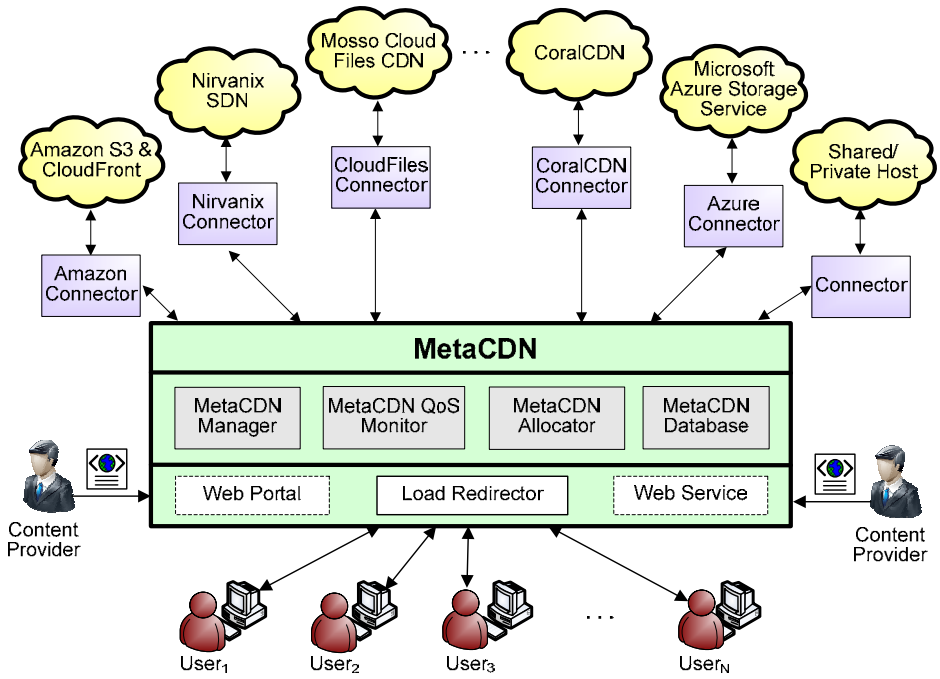


Figure 7.1: Components of the MetaCDN overlay system.

7.3 The MetaCDN Overlay

MetaCDN is developed as a simple, general purpose, and reusable overlay network in the face of daunting challenges faced by content providers to exploit the resources of multiple storage clouds. It provides a platform to harness content delivery services, by hiding the complexity of using unique Web services or programmer APIs coupled with each cloud provider. End-users experience little of the complex technologies associated with MetaCDN. Content providers interact with the service in a limited number of ways, such as enabling their content to be served, viewing traffic reports, and receiving usage-based billing. In this section, we provide an overview of the MetaCDN system focusing on its existing capabilities, high level system characteristics and perform a comparative analysis to related systems.

7.3.1 Overview

MetaCDN has opened up opportunities for the content providers and end-users to reap rewards through low-cost, high performance and easy to use distributed CDN. Figure 7.1 provides an illustration of the MetaCDN system. It is coupled with each

storage cloud via *connectors*, which provide an abstraction to conceal different access methodologies to heterogeneous providers. These connectors (cloud provider specific; and FTP, SSH/SCP or WebDAV for shared or private hosts) provide basic operations for creation, deletion, rename, and listing of replicated content. End-users can access the MetaCDN overlay either through a Web portal or via RESTful Web services. In the first case, the Web portal acts as an entry point to the system and performs application level load balancing for end-users who intend to download content that has been deployed through MetaCDN. Content providers can sign up for an account on the MetaCDN system and enter credentials for any storage cloud providers that have an account with. Upon authentication, they can utilize MetaCDN functionalities to intelligently deploy content over geographically spanned replicas from multiple storage clouds, according to their performance requirements and budget limitations.

A distributed MetaCDN gateway (middleware entity) provides the logic and management required to encapsulate the functionality of upstream storage cloud providers with a number of core components. The *MetaCDN Allocator* performs optimal provider selection and physical content deployment using four options, namely, *maximize-coverage*, *geolocation-based*, *cost-optimized*, and *QoS-optimized* deployment. The *MetaCDN QoS Monitor* tracks the current and historical performance of participating storage providers. The *MetaCDN Manager* has the authority on each content provider's current deployment and performs various housekeeping tasks. The *MetaCDN Database* stores crucial information, such as user accounts and deployments, and the capabilities, pricing, and historical performance of providers. Finally, the *MetaCDN Load Redirector* is charged with different redirection policies and is responsible for directing users to the most appropriate replica according to performance requirements. We refer to a prior work [29] for details on critical MetaCDN functionalities along with a full architectural description and the development methodology.

7.3.2 System Characteristics

MetaCDN is a smart, agile and flexible approach for content delivery that is willing to break with tradition. Specifically, the following set of attributes can be used to characterize it:

- ***Multi-tenancy.*** MetaCDN provides content delivery services for many content providers and end-users on the same distributed infrastructure for different content types. With a cloud-based model, all resources and costs are shared among a large pool of users, enabling genuine savings and economics of scale.
- ***Elasticity.*** It is able to support diverse range of performance requirements from content providers and end-users. This characteristic allows it to quickly and gracefully respond to high request rates at reasonable response time.
- ***Scalability.*** MetaCDN resources are dynamically scalable to handle workload variations with growing number of content providers and end-users, thus enabling optimum resource utilization.
- ***Load sharing.*** It offers automatic and totally transparent load balancing on end-user requests. It enables faster absorption of load spikes with the aid of different load balancing and redirection policies.
- ***Global availability and reliability.*** MetaCDN is a truely on-demand service to provide content delivery functionalities to all authorized users from anywhere on the Internet natively. It has the ability to automatically avoid failed replicas or replicas without desired content. In particular, its tolerance to high failure rate ensures that end-users suffer from little to no outages (i.e. rare infrequent downtime), such as server or network failures.
- ***Ease of use/operability.*** It can be accessed through a simple Web interface that mimics the look and feel of familiar consumer Web applications, making it extremely intuitive and easy to operate.
- ***Reusability and cost of development.*** The low development cost of using storage clouds for MetaCDN ensures significantly reduced upfront costs. It is implemented by means of reusable simple APIs exposed by the storage cloud providers, while avoiding too many parameters that must be tuned in order to perceive good performance for diverse content providers and their end-users.
- ***Metered services.*** By using the third-party content delivery services of the MetaCDN system, content providers have to pay only for the capacity that

they use from upstream cloud providers. Usage information for each replica (e.g. download count and last access) is recorded in order to track the cost incurred for specific content from a content provider.

- ***Security.*** It addresses crucial security concerns, as content providers use their own credentials for any cloud storage or other provider they have an account with. Thus, it allows content providers to entrust their content to MetaCDN for processing, rest assuring that it will be protected from theft, loss or corruption.

7.3.3 A Comparative Analysis

MetaCDN complements CDN peering by providing an end-to-end cloud-based solution, coupled with on-demand intelligent request-redirection. In this section, we first ascertain MetaCDN's feasibility and position it as a distributed CDN by presenting a comparative study with related systems. Then we study existing redirection mechanisms available in literature and used in practice. While some of these systems and techniques have been mentioned previously (Chapter 3), we briefly revisit them to endorse MetaCDN's novelty and uniqueness.

MetaCDN and related Systems. The Content Distribution Internetworking (CDI) [72] model lays the foundation for interconnecting providers. Following the footsteps of the CDI initiative, several research efforts explore the benefits of internetworking/peering of CDN providers, content providers, Peer-to-Peer (P2P) networks, and overlays with main focus on offering increased capacity, intelligent server selection, reduced cost, and improved fault tolerance. Examples include CDI protocol architecture [209, 210], multi-provider peering [10], Synergy overlay internetworking [119], peer-assisted content delivery [208], group-based content delivery [129], provisioning content delivery over shared infrastructure [142], use of emerging technologies for the development of enhanced content delivery service [84], resource management in a Grid-based CDN [74], capacity provisioning networks [93], open CDN implementation [139], and our work, i.e. CDN peering [161, 162]. In contrast, MetaCDN assumes no cooperation or peering. Rather it follows a brokering-based approach (Chapter 3: Section 3.4.1) as in CDN brokering [24], which is a content delivery bro-

kerage system deployed on the Internet on a provisional basis. MetaCDN differs in that it functions as a Content Delivery Cloud [60, 154, 157], replicating content over its distributed infrastructure spanning multiple continents, and providing content delivery services to far flung end-users. It has demonstrated improved content delivery performance, and enumerate its content-serving utility and content provider's benefits from using it [29, 157]. While MetaCDN is comparable to the collaborative CDNs, such as CoDeeN [223], CoralCDN [87], and Globule [167, 168], it is significantly different as it integrates storage cloud resources spanning the globe to provide content delivery services.

Many Websites have utilized individual storage clouds to deliver some or all of their content [78], most notably the New York Times [95] and SmugMug [131]. On the contrary, MetaCDN provides general purpose reusable content delivery services by interacting and leveraging multiple cloud providers. MetaCDN is positioned as a logical fit in the industry initiatives to couple content delivery capabilities with existing cloud deployments, such as Amazon S3 and CloudFront; Silverlining and VoxCAST CDN; Mosso Cloud Files; Nirvanix SDN, which partners with CDNetworks for content delivery; TinyCDN, which leverages Amazon Web services and cloud computing; and Edge Content Network (ECN) from Microsoft, which is reported to partner with Limelight Networks for content delivery [135]. However, as these systems use centralized or a small number of datacenters, they may suffer from deteriorated end-user experience due to network congestions, peering point congestion, routing inefficiencies, and other bottlenecks of the Internet middle mile [124]. On the contrary, MetaCDN is attributed with a distributed CDN infrastructure to overcome the challenges posed by the Internet's middle mile and ensure that end-user performance does not fall short of expectations. The MetaCDN approach is analogous to the Akamai cloud computing initiative [124], which provides cloud optimization services for its highly distributed EdgePlatform. However, unlike Akamai it endeavors to achieve true economics of scale by exploiting the *pay-as-you-go* model of upstream cloud providers.

Recent innovations such as P4P [226] and its companion traffic engineering mod-

els [103] enable P2P to communicate with network providers through a portal for cooperative content delivery. Such proactive network provider participation optimizes global peer-to-peer connections as it saves significant user costs, and by using local connections also speeds up download times for P2P downloaders by 45%. MetaCDN endorses them in the sense that it assists toward a systematic understanding and practical realization of the interactions between storage clouds, which provide an operational storage network and content delivery resources, and content providers, who generate and distribute content.

Request-redirection techniques. Request-redirection is generally used to direct end-user requests to replica servers based on various policies and a possible set of metrics, such as network proximity, user perceived latency, bandwidth, content availability and replica server load. There exist multiple request-redirection mechanisms, which can be categorized in a number of ways according to different performance objectives.

Barbir et al. [19] categorize the known request-redirection techniques in CDNs into *DNS-based*, *transport-layer* and *application-layer* redirection. In DNS-based techniques, a specialized DNS server is augmented in the name resolution process to return different server addresses to end-users. They are the most common due to the ubiquity of the DNS system as a directory service. The performance and effectiveness of DNS-based redirection techniques have been studied in a number of recent studies [24, 133, 192]. Despite its wide usage, DNS-based approaches are found to suffer from the following drawbacks: (a) actual end-user request is not redirected, rather its Local DNS (LDNS), assuming that end-users are near to their LDNS; (b) browser's request is cached due to the hierarchical organization of the DNS service; (c) the DNS system is not designed for very dynamic changes in the mapping between hostnames and IP addresses; and (d) most significantly DNS can not be relied upon as it can have control over as little as 5% of incoming requests in many instances [41]. In transport-layer redirection, the information available in the first packet of the end-user request, in combination with user-defined policies and other metrics are used to take redirection decision. Several research [128, 150, 227] report using this

approach for redirection. In general, this approach is used in combination with DNS-based techniques. While this approach is suitable for steering end-users away from overloaded replica servers, the associated overhead limits its usage for long-lived sessions such as FTP and RTSP. Finally, application-layer redirection involves deeper examination of end-user request packet to provide fine-grain redirection. However, this approach may suffer from the lack of transparency and additional latency. URL rewriting and HTTP 302 redirection are the examples of techniques using this approach. In the context of MetaCDN, the system exploits a combination of DNS-based and application-layer techniques for request-redirection. Specifically, name resolution for the base MetaCDN URL is performed using DNS-redirection and end-user request for specific content (Web object) is serviced using application-layer redirection. This aspect is elaborated in Section 7.5.2.

With the objective to minimize Web access latency, request-redirection can be partitioned into client and server-side techniques. Client-side redirections in CDNs [64, 109, 181, 222] are based on the premise that the network is the primary bottleneck. They tend not to rely on any centralization as redirections occur independently. Server-side techniques perform URL redirection using HTTP status code. They direct all incoming requests to a set of clustered hosts based on load characteristics. These techniques are mainly application specific and more suited for clustered servers. There also exist significant research [45, 46, 110, 177] combining client and server-side redirection. This hybrid approach works well when the bottleneck is not clearly identified or varying over time. According to this categorization, MetaCDN complements the hybrid request-redirection technique; by performing server-side gateway redirection and client-side HTTP 302 redirection for content requests.

In terms of content retrieval, request-redirection techniques can be divided into *full* and *selective* (or *partial*) redirection. In full redirection, the DNS server is modified in such a way that all end-user requests are directed to a replica server. This scheme requires that either replica servers hold all the content from the origin server, or that they act as surrogate proxies for the origin server. On the other hand, in selective redirection, a content provider modifies its content so that links to specific embedded

Web objects have host names in a domain for which the CDN provider is authoritative. Thus, the base HTML page is retrieved from the origin server, while embedded objects are retrieved from CDN replica servers. While full replication has dynamic adaptability to new hot-spots, it is not feasible considering the on-going increase in Web objects size. A selective redirection works better in the sense that it reduces load on the origin server and on the Web site's content generation infrastructure. Moreover, if the embedded content changes infrequently, it exhibits better performance. While it is possible to use the MetaCDN replica infrastructure to enable full redirection, we limit our work for selective redirection by storing only embedded Web content into replicas and directing end-user requests to them.

Request-redirection mechanisms are governed by policies that outline the actual redirection algorithm on how to perform server selection in response to an end-user request. These policies can be either *adaptive* or *non-adaptive*. Adaptive policies consider the current system condition, whereas non-adaptive policies use some heuristics in order to perform target server selection. The literature on request-redirection policies is too vast to cite here (see the survey by Sivasubramanian et al. [198] and the references therein for initial pointers for redirection policies in CDN context). MetaCDN deploys adaptive redirection with the ability to cope with degenerated load situations. In particular, it strives to demonstrate high system robustness in the face of unanticipated events, e.g. flash crowds.

There exist significant research efforts [9, 79, 173, 182] that model request-redirection as a mathematical problem. They attempt to find a solution from an operations research perspective by modeling redirection as a graph theory, optimization, delay constrained routing, or server assignment problem. Most of these work use simulations to evaluate the performance of their approach. On the contrary, the MetaCDN redirection is evaluated through a proof-of-concept implementation on its distributed infrastructure.

We also draw similarity with the request-redirection techniques used in the collaborative CDNs, e.g. CoDeeN [223], CoralCDN [87], Globule [167, 168], and PRSync [191], which perform overlay redirection by exploiting request locality, network measurement, topology, and AS-based proximity. Our work is in line with them as

MetaCDN request-redirection is based on metrics such as geographic proximity, cost, request traffic, and QoS metrics (response time, throughput, HTTP response code). Our uniqueness lies in adding the capability for quantifying traffic activities using a network *utility* metric within MetaCDN while intelligently redirecting user requests.

7.4 Maximizing MetaCDN Utility

To ensure high performance content delivery via MetaCDN, we aim at improving its content-serving utility. To achieve this, similar to the utility of CDN peering (Chapter 6), we quantify MetaCDN utility based on its traffic activities and represent the usefuleness of its replica infrastructure in terms of data circulation in its distributed testbed. We first formulate the utility maximization problem with quantitative expressions. Then in Section 7.5, we devise a utility-based request-redirection policy.

7.4.1 Problem Formulation

We use $R = \{r_i\}$, $i \in \{1, 2, \ldots, M\}$ to denote the set of end-user requests, with r_i being the i-th arriving request to the MetaCDN overlay system, comprising a set of N replicas. Utility maximization in MetaCDN can be achieved from two perspectives. The first aspect is the profit maximization of MetaCDN, which is formulated as:

$$\text{maximize} \quad \sum_{r_i \in R, j \in N} U_{mcdn} x_{ij} \quad (Profit) \tag{7.1}$$

where U_{mcdn} is content-serving utility of the system; indicator variable x_{ij}=1 if MetaCDN replica j serves request r_i within service requirements; and x_{ij}=0 otherwise.

The second aspect examines the general welfare of the content provider for using the MetaCDN infrastructure to maximize its own benefit (surplus). Each content provider obtains a perceived utility U_{CP} (benefits) for QoS-constrained content delivery to its end-users via MetaCDN. The measured utility is expressed as the fraction of processed requests (throughput) or the total valuation (weighted throughput). It can be formulated either by maximizing the following one or two measurements:

$$\text{maximize} \quad \sum_{r_i \in R, j \in N} x_{ij} \quad (Throughput) \tag{7.2}$$

$$\text{maximize} \sum_{r_i \in R, j \in N} U_{CP} x_{ij} \quad (\textit{Weighted throughput}) \tag{7.3}$$

where U_{CP} is the content provider's perceived utility.

7.4.2 Quantitative Expressions

We now derive the quantitative measure for MetaCDN utility U_{mcdn} and content providers perceived utility U_{CP}. Ideally for the MetaCDN overlay system, the most useful replicas are those exhibiting the highest utility. We follow a similar approach as in Chapter 6 to quantify utility. However, we redefine the utility metric as we are working with a practical system where it is possible to capture the true traffic activities, in terms of the number of bytes transferred during content replication and servicing. Hence, utility in MetaCDN is expressed with a value in [0, 1] that reflects the relation between the number of bytes of the served content against the number of bytes of the replicated content. Formally, utility of a MetaCDN replica *i* is:

$$u_i = (2/\pi) \times \arctan(\xi) \tag{7.4}$$

The main idea behind this metric is that a replica is considered to be useful (high utility) if it serves content more than it replicates, and vice versa. The parameter ξ is the ratio of the serviced bytes to the replicated bytes, i.e.

$$\xi = \textit{No of bytes serviced} / \textit{No of bytes replicated} \tag{7.5}$$

The resulting utility from (7.4) ranges in [0, 1]. The value u_i=1 is achieved if the replica only serves content, without replicating (ξ=infinity). It results when a replica already has the content, and does not replicate a new copy of the content for serving successive requests. On the contrary, the value u_i=0 is achieved if the replica has the content, however fails to serve (ξ=0) due to service over-provisioning and/or network perturbations under heavy traffic surges. The content-serving utility U_{mcdn} of MetaCDN can be expressed as a mean value of the individual replica utilities, i.e.

$$U_{mcdn} = \sum_{n=1}^{N} u_i \Big/ N \tag{7.6}$$

MetaCDN outsources customer's (content provider) content to the replicas and it is charged by the cloud providers based on usage. Since the utility measure captures

the usage of storage cloud resources, the measured value can be easily translated into a price of the offered services. The resulting price can be used to derive a content provider's benefits or perceived utility. We draw inspiration from Hosanagar et al. [99], who show that the benefits of a content provider depends on its revenue, benefit from content delivery to its end-users through a CDN, replication cost, and usage-based charges. We adopt this approach by using performance measures of end-users that belong to a content provider. We gauge the throughput for serving content requests and the response time improvement by using MetaCDN over direct replica access. We also measure the replication cost and interpret the pricing of storage clouds according to the MetaCDN utility. We express a content provider's perceived utility as:

$$U_{CP} = T(X) + (R - R_d) \times X - \psi b - P(u) \quad \textbf{(7.7)}$$

where $T()$ is the weighted throughput for the traffic volume of X requests; R and R_d respectively are the perceived response times from direct replica access and via MetaCDN; ψ is the unit replication cost; b is the content size; and $P(u)$ is the utility-based pricing function.

7.5 Request-Redirection Design

An efficient request-redirection technique is vital to extend the reach and scale of MetaCDN. In this section, we analyze the design space of a competent redirection technique and describe the proposed utility-based request-redirection technique.

7.5.1 Design Space

Designing a request-redirection strategy that does not sacrifice the scalability, transparency, availability and performance benefits of cloud-based distributed content delivery service, i.e. MetaCDN, is a challenging task. A candidate redirection technique should have the following properties:

- ***Scalability.*** It should be responsive to changing circumstances. It should aid the system with the ability to gracefully scale and expand its network reach in

order to handle new and large number of data, end-user requests, and transactions without any significant decline in performance.

- ***Load balancing.*** With the aid of the redirection technique, MetaCDN as a service provider should be able to effectively react to overload conditions by selecting least loaded optimal server(s) for serving content requests. The load balancing decisions should ensure that end-users experience reasonable content delivery performance.
- ***Distributed redirection.*** It should not rely on any centralization and all redirectors (i.e. MetaCDN gateway) should operate independently. It should also accommodate any dynamic changes in network performance and incoming request traffic.
- ***Transparent name resolution.*** DNS mapping during redirection should be transparent to end-users. In order to transparently contact a replica server for desired content, redirection should ensure a *one-to-many* mapping from the hostname to one of the IP addresses of distributed replicas.
- ***Fault transparency.*** It should ensure that unresponsive replicas are detected, bypassed and end-users are unaware of the redirection to other replicas. Moreover, previously failed replicas that become available again should be incorporated quickly.
- ***Flexibility.*** There should be provision to accommodate different request-redirection techniques to provide options to content providers and its users with varied objectives. In addition, a candidate request-redirection technique should improve the usefulness of distributed replicas.
- ***Server decoupling.*** The redirection logic should be implemented without any change of the existing client or server code, conforming to existing standards. It should also be possible to deploy the devised redirection scheme easily, preferrably as a plug-in to the server, with minimum effort. Thus, it should be ensured that the implementation overhead of a given request-redirection technique is minimal.

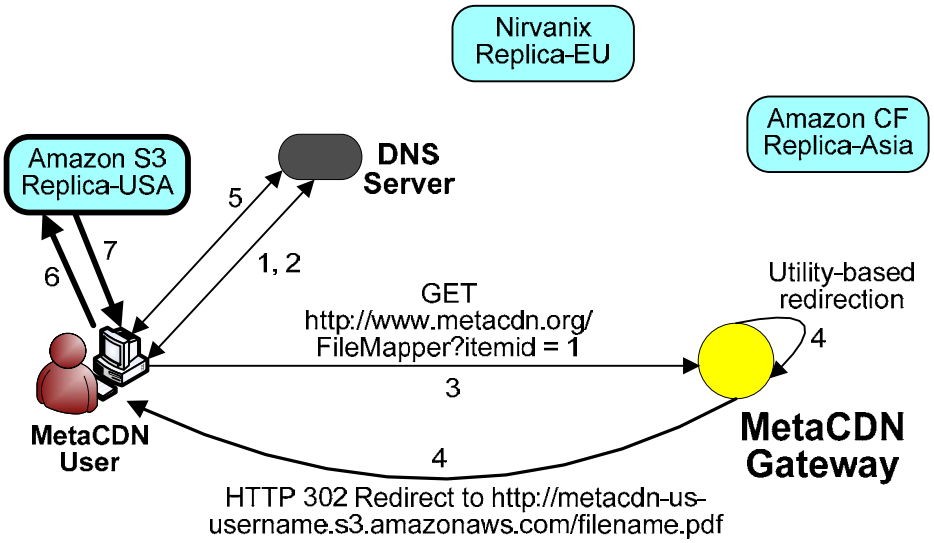


Figure 7.2: Utility-based request-redirection in MetaCDN.

7.5.2 Utility-based Request-Redirection

Request-redirection in MetaCDN takes place under the governance of the MetaCDN gateways, which resemble distributed request-redirectors to forward end-user content requests to appropriate replica server. The MetaCDN gateway is capable of utilizing any request-redirection technique that is plugged into the MetaCDN Load Redirection module. Integrating a new request-redirection scheme does not require any changes to the server or client-side.

To choose the optimal server for an end-user request, the MetaCDN Load Redirector module evaluates the utility metric reflecting the state of its replicas and the network conditions between end-users and replica sites. The measured utility is used for a utility-based request-redirection policy (Figure 7.2) for MetaCDN to serve data-rich content, thus improving the content delivery performance by relieving network congestions.

As shown in Figure 7.2, the sequence of steps for an end-user in the East Coast of the USA to retrieve content through MetaCDN is as follows:

1) The end-user issues an HTTP request for a content that has been deployed by the MetaCDN Allocator using one of the content deployment options avail-

able (Section 7.3.1). The browser attempts to resolve the base hostname (*www.metacdn.org*) for the MetaCDN URL *http://www.metacdn.org/FileMapper?itemid=XX*, where *XX* in the URL format is a unique key associated with the deployed content.

2) The Local DNS (LDNS) of the end-user contacts the authoritative DNS (ADNS) for that domain to resolve this request to the IP address of the closest MetaCDN gateway, e.g. *http://us.metacdn.org*.

3) The end-user (or its browser) then makes an HTTP GET request for the desired content on the MetaCDN gateway.

4) In the case of utility-based request-redirection, the MetaCDN Load Redirector is triggered to select the highest-utility optimal replica that conforms to the specified service requirements. At this point, the MetaCDN gateway returns an HTTP redirect request with the URL of the selected replica. The following tests are performed to determine the best replica for serving user requests:
   - Is there a content replica available within required response time threshold?
   - Is the throughput of the target replica within tolerance?
   - Is the end-user located in the same geographical region as the target replica?
   - Is one of the target replicas preferred, according to user requirements or any administrative settings?
   - Is the replica utility the highest among all target sites?
   - If there is more than one replica with the same highest utility, which replica site provides the fastest response time to be the selected replica?

5) Upon receiving the URL of the selected replica, the DNS resolves its domain name and returns the associated IP address to the end-user.

6) The user sends request for the content to the selected replica.

7) The selected replica satisfies the user request by serving the desired content.

If it is assumed that all candidate replicas are available and have capacity, i.e. response time and throughput thresholds are met, the MetaCDN system checks for the

continent/geographic location and administrative preference (an indicative flag used by MetaCDN manager to manually prefer or avoid a replica).

MetaCDN achieves transparency as end-user browsers automatically access the redirection service, being redirected by the MetaCDN gateway. End-users have least possible to do to take benefit of request-redirection. They see only MetaCDN URL and they have no way for discovering the address of a replica when using the redirection service and accessing the replica server directly. Thus, we prevent an end-user to keep an explicit reference to a replica, which may cause dangling pointers during the downtime of the replica.

While MetaCDN Load Redirector ensures directing users to the best responding replica, an extra feature is realized through its ability to automatically avoid failed replicas or replicas without the desired content. Bypassing occurs in the following two ways. Firstly, if a replica has the desired content, but shows limited serving capacity due to network congestions, it is reflected in its measured network utility metric, exhibiting a low value. As a consequence, the replica is not considered as a candidate for redirection. Secondly, if the replica does not have the desired content, it can not serve end-user requests and thus leads to an insignificant utility value. Hence, it is automatically discarded to be considered as a candidate replica. In addition, a secondary level of internal redirection enabled by an individual cloud provider ensures that request-redirection does not overload any particular replica.

7.6 Evaluation Methodology

We conduct a *proof-of-concept* testbed experiment to determine the content delivery utility of MetaCDN, evaluate the performance of the utility-based request-redirection policy, and measure the user perceived response time and throughput. Figure 7.3 provides a schematic representation of the experimental testbed and Table 2.1 provides a summary of the conducted experiment. The global MetaCDN testbed spans six continents with distributed clients at different institutions; replicas from multiple storage cloud providers; and MetaCDN gateways, hosted on the Amazon Elastic Computing Cloud (EC2) and a cluster at the University of Melbourne, Australia.

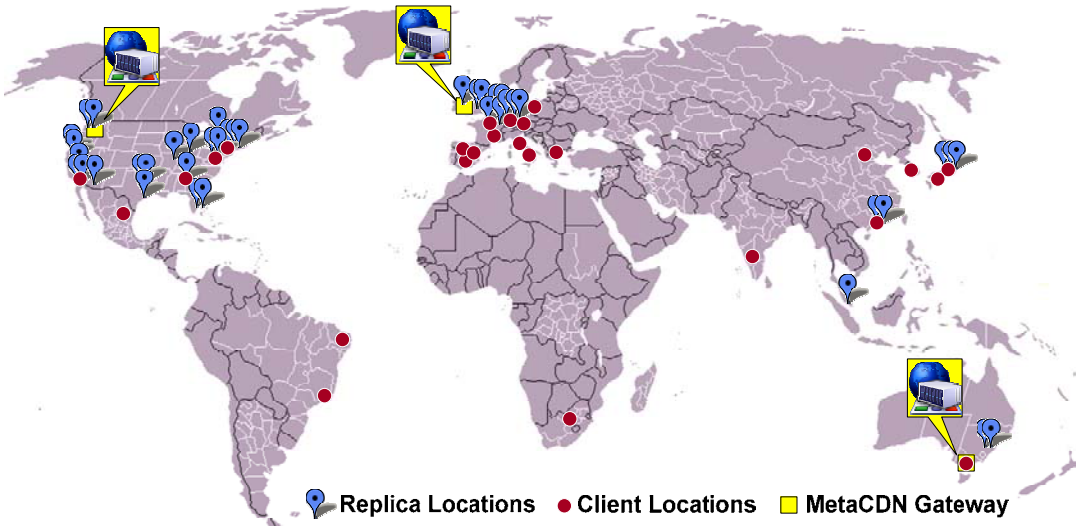


Figure 7.3: Experiment testbed.

Table 7.1: Summary of the experiment.

	Category	Value	Provider	Locations
Experiment Testbed	Number of MetaCDN gateways	3	Amazon EC2 and own cluster	Asia/Australia, Europe, and North America
	Number of replicas	40	Amazon, Mosso and Nirvanix	Asia, Australia, Europe, and North America
	Number of clients (end-user nodes)	26	Voluntary	Asia, Australia, Europe, North and South America, and Africa

	Category	Description
Experiment Details	Total experiment time	48 hours
	Duration of an epoch	2 hours
	Maximum user requests/epoch	30 requests from each client
	Service timeout for each request	30 seconds
	Test file size	1 KB and 5 MB
	Content Deployment	Maximize-coverage deployment
	Request-redirection policies	Random, Geo, and Utility

	Category	Distribution	PMF	Parameters
End-user Request Modeling	Session inter-arrival time [81]	Exponential	$\lambda e^{-\lambda x}$	$\lambda = 0.05$
	Content requests per session [14]	Inverse Gaussian	$\sqrt{\frac{\lambda}{2\pi x^3}} e^{\frac{-\lambda(x-\mu)^2}{2\mu^2 x}}$	$\mu = 3.86$ $\lambda = 9.46$
	User think time [21]	Pareto	$\alpha k^{\alpha} x^{-\alpha-1}$	$\alpha = 1.4$, $k = 1$

All client locations, except in Africa, South America and South Asia, have high speed connectivity to major Internet backbones to minimize the client being the bottleneck during experiments. We used test files of size 1KB and 5MB, deployed by the

MetaCDN Allocator module, which was instructed to maximize coverage and performance, and consequently the test files were deployed in all available replica locations of the storage cloud providers integrated to MetaCDN. While these file sizes are appropriate for our experiments, a few constraints restrict us to use varied and/or even larger sized files. Firstly, the experiments generate heavy network traffic consuming significant network bandwidth, thus larger file trafficking would impose more strain and network congestions on the voluntary clients, which some clients may not be able to handle. Moreover, at some client locations, e.g. India and South Africa, Internet is at a premium and there are checks regarding Internet traffic so that other users in the client domain accessing the Internet are not affected.

The experiment was run simultaneously at each client location over a period of 48 hours, during the middle of the week in May 2009. As it spans two days, localized peak times (*time-of-day*) is experienced in each geographical region. The workload to drive the experiment incorporates recent results on Web characterization [14, 21, 81]. The high variability and self-similar nature of Web access load is modeled through heavy-tailed distributions. The experiment time comprises epochs of 2 hours, with each epoch consisting of a set of user sessions. Each session opens a persistent HTTP connection to MetaCDN and each client generates requests to it to download each test files, with a timeout of 30 seconds. Between two requests, a user waits for a *think time* before the next request is generated. The mean think time, together with number of users defines the mean request arrival rate to MetaCDN. For statistical significance, each client is bounded to generate a maximum number of 30 requests in each epoch. The files are downloaded using the UNIX utility, *wget*, with the *--no-cache* and *--no-dns-cache* options to ensure that a fresh copy of the content is downloaded each time (not from any intermediary cache) and that the DNS lookup is not cached either.

7.6.1 Schemes and Metrics for Comparison

The primary objectives are to measure MetaCDN utility, evaluate performance of the proposed utility-based request-redirection policy, and provide observations on how MetaCDN's content serving ability is varied during the experiment. For performance comparison, we experiment with two other request-redirection policies that are pre-

sent inside the MetaCDN system, as described in the following:

- ***Random redirection.*** It is a simple baseline policy where each content request is sent to a randomly picked replica. We use this scheme for comparison purpose to determine a reasonable level of performance, since we expect our approach to scale with the increasing number of clients and MetaCDN replicas, and to not exhibit any pathological behavior due to the assignment patterns. The drawback of this approach is to often increase latency by not picking up the most appropriate replica. Moreover, adding more servers does not reduce the working set of each server.
- ***Geolocation-based redirection.*** It exploits the request locality by taking into account end-user preferences and directing the user to the closest physical replica in the specified region(s). For this purpose, a geolocation service is utilized that finds the geographic location (latitude and longitude) of the end-user and measures their distance from each matching replica using a simple spherical law of cosines, or a more accurate approach such as the Vincenty formula for distance between two latitude/longitude points [220], to find the closest replica. Although there exists a strong correlation between the performance experienced by end-users and their locality to replicas [29], there is no guarantee that the closest replica is always the best choice, due to cyclical and transient load fluctuations in the network path.

We measure the *response time* and *throughput* obtained from each client location. The first performance metric captures the end-to-end performance for end-users when downloading a 1 KB test file from MetaCDN. Due to the negligible file size, the response time is dominated by DNS lookup and HTTP connection establishment time. Lower value of response time indicates fast serviced content. The latter metric shows the transfer speed obtained when the 5 MB test file is downloaded by the users from the MetaCDN replica infrastructure. It provides an indication of consistency and variability of throughput over time.

The *utility* of MetaCDN is measured according to the quantitative expressions in Section 7.4. A high utility value shows the content-serving ability of the system, and

signifies its durability under highly variable traffic activities. To emphasize the impact of request-redirection on the measured utility, we use the *probability* that MetaCDN achieves a given level of utility as the performance metric. Finally, based on the measured observations, we determine the benefits of a content provider (*surplus*) from using the MetaCDN system. Table 7.2 summarizes the performance indices used in the experimental evaluation.

Table 7.2: List of performance indices.

Performance Index	Description
Response time	The time experienced by an end-user to get serviced
Throughput	Transfer speed to download a test file by an end-user
Utility	Content-serving ability, ranges in [0, 1]
Prob(Utility achieved)	The probability or the fraction of time that the system achieves the given utility
Content provider's benefit (Surplus)	Surplus from using MetaCDN, expressed as a percentage

7.7 Empirical Results

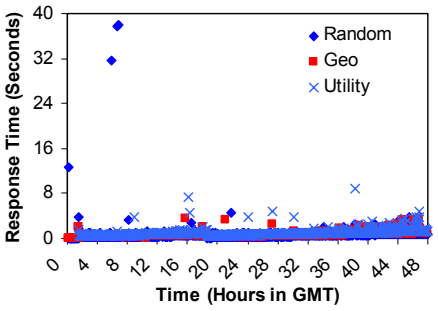
To avoid redundancy, we present results from the following eight representative client locations in five continents—Paris (France), Innsbruck (Austria), and Poznan (Poland) in Europe; Beijing (China) and Melbourne (Australia) in Asia/Australia; Atlanta, GA, and Irvine, CA (USA) in North America, and Rio de Janeiro (Brazil) in South America.

7.7.1 Response Time Observations

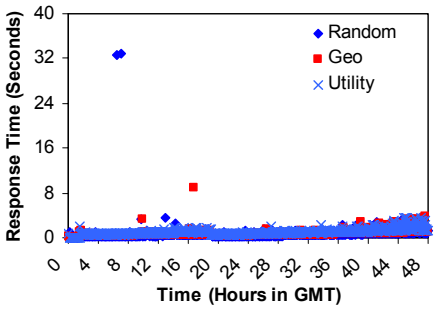
Figure 7.5 shows the end-to-end response time experienced by end-users when downloading the 1 KB test file over a period of 48 hours. The measure of the response time depends on the network proximity, congestions in network path and traffic load on the target replica server. It provides an indication of the responsiveness of the replica infrastructure and the network conditions in the path between the client and the target replica which serves the end-user. We observe a general trend that the clients experience mostly consistent end-to-end response time. For all the request-redirection policies, the average response time in all the client locations except Beijing is just over 1 second, with a few exceptions. Notably the users in Beijing

experience close to 4 seconds average response time from the MetaCDN infrastructure. This exception originates as a consequence of firewall policies applied by the Chinese government. Similar observations have been reported in a previous measurement study [179], which demonstrates that the failure characteristics on the Internet path to the edge nodes in China are remarkably different than the Internet paths to the edge nodes in other part of the world.

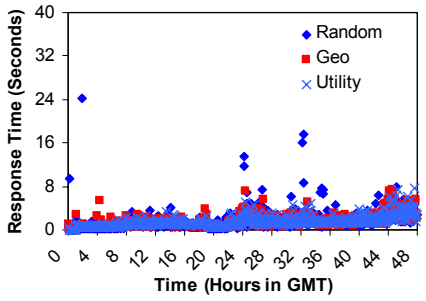
At several time instances during the experiment, end-users experience increased response time. The resulting spikes are due to the sudden increases in request traffic, imposing strain on the MetaCDN replicas. Under traffic surges, the MetaCDN Load Redirector module activates to handle peak loads. As a consequence, end-user requests are often redirected to a target replica outside its authoritative domain and/or are served from an optimal distant proximity server, thereby, contributing to the increased response time. However, MetaCDN handles peak loads well to provide satisfactory service responsiveness to end-users. This phenomenon of increased response time is more visible for random-redirection. As it makes a random choice, often the target replica selection is not optimized, thus leading to highly variable response time. Especially, at several occasions, users observe more than 30 seconds response time, thus leading to service timeout. Geo-redirection directs user requests to the closest proximity server, understandably producing low response time. On the contrary, utility-redirection chooses the highest utility replica, which may not be in close proximity to an individual client location. Nevertheless, we do not find a clear winner between them in terms of response time, as they exhibit changeable performance at different client locations. Utility-redirection performs as well as geo-redirection as we observe similar performance in all clients except Paris and Melbourne. End-users in Paris enjoy better average response time (0.77 seconds) with geo-redirection, due to their close proximity to the Amazon, Mosso and Nirvanix nodes in Frankfurt (Germany), Dublin (Ireland), and London (UK). For Melbourne, the reason behind better performance of geo-redirection is the existence of the Mosso node in Sydney. For both of these two clients, utility-redirection policy directs requests to a distant replica than the closest one and results in increased response time.



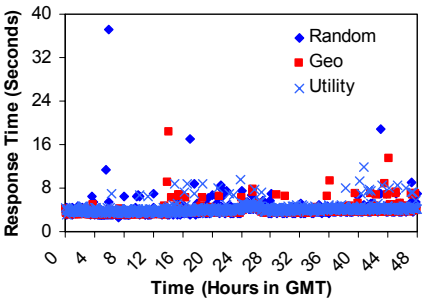
(a) Paris



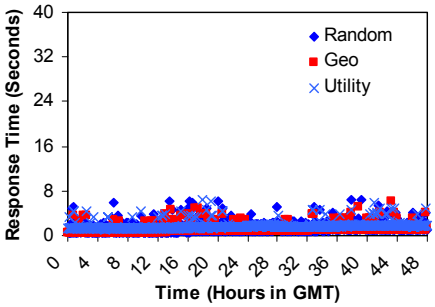
(b) Innsbruck



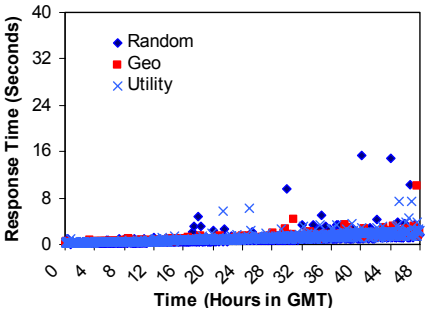
(c) Poznan



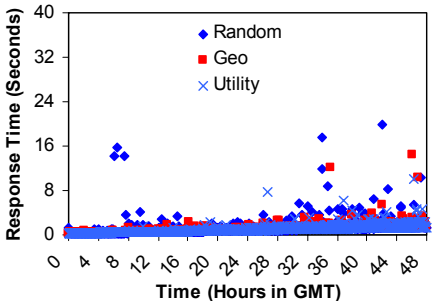
(d) Beijing



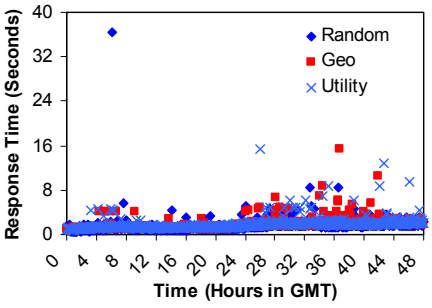
(e) Melbourne



(f) Atlanta

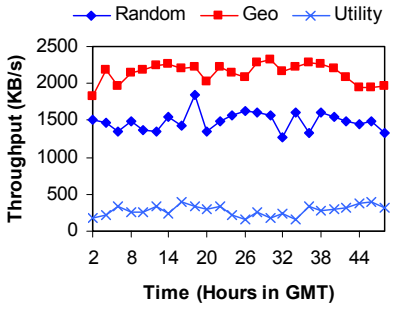


(g) Irvine

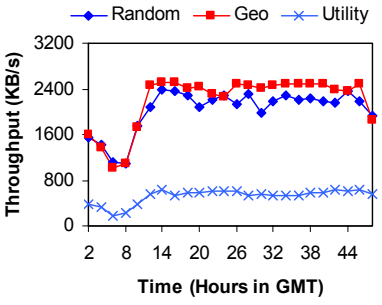


(h) Rio de Janeiro

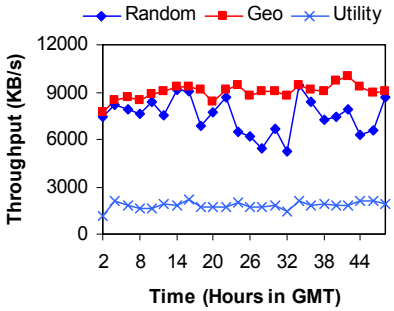
Figure 7.4: Response time obtained in each client location.



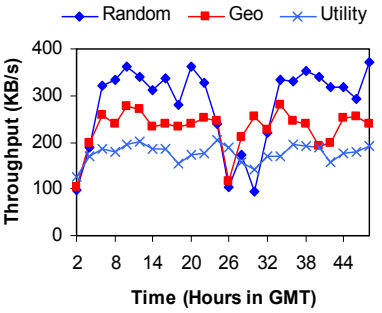
(a) Paris



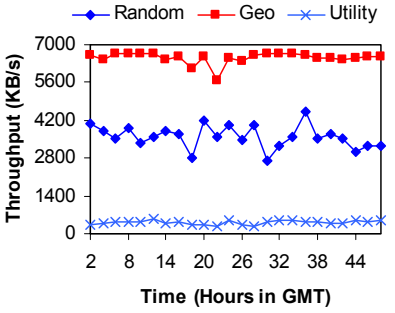
(b) Innsbruck



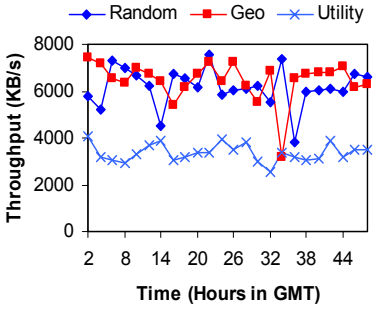
(c) Poznan



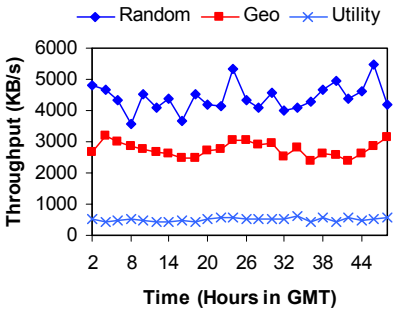
(d) Beijing



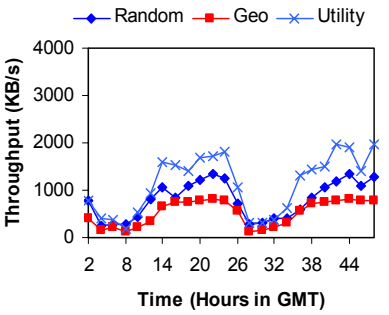
(e) Melbourne



(f) Atlanta



(g) Irvine



(h) Rio de Janeiro

Figure 7.5: Average throughput obtained in each client location.

7.7.2 Throughput Observations

Figure 7.5 shows the average throughput obtained per two hours, when downloading content (5MB file) via MetaCDN. As expected, we observe that in almost all the client locations, geo-redirection results in highest throughput as the users get serviced from the closest proximity replica. However, it performs worse than random-redirection for the Irvine client. The reason is that random-redirection decision in this location most of the time selects close proximity Amazon replica(s) with better network path than that of geo-redirection, which chooses Mosso replica. Moreover, the service capability from these two replicas and the network path between the replica and client also contribute to the observed throughput variations.

For most of the clients, except Rio de Janeiro, utility redirection performs much worse than geo-redirection. The reason is understandable, as utility-redirection emphasizes maximizing MetaCDN's utility rather than serving an individual user, thus sacrificing end-user perceived performance. For Rio de Janeiro, geo-redirection leads to the closest Mosso node in the USA, whereas utility-redirection results in more utility-aware replica, which is the Amazon node(s) in the USA. It could be presumed that Amazon node supersedes the Mosso node in terms of its service capability, better network path, internal overlay routing, and less request traffic strain.

It is observed that users in Poznan enjoy the best average throughput, which is 9MB/s for geo-redirection. The reason is that the client machine is in a MAN network, which is connected to the country-wide Polish optical network PIONEER with high capacity channels dedicated to the content delivery traffic. Another client location with high throughput is Atlanta, which achieves speeds of approximately 6.2 MB/s for geo-redirection and 3.3 MB/s for utility-redirection, due to the existence of better network path between the client and the MetaCDN replica infrastructure. This reasoning is deemed valid, since there are Mosso nodes in the same location.

Alike response time, end-users in China achieves the lowest throughput among all the client locations. The underlying reason is again checks on the request traffic and bandwidth constraints due to firewall policies. We put more emphasis on the results from Melbourne, which is of interest as Australia is not as highly connected as

Europe or North America, depending on a small number of expensive international links to major data centers in Europe and the USA. We observe that due to the existence of a nearby Mosso node in Sydney, the users in Melbourne experience 6.5 MB/s of throughput with geo-redirection and 3.6 MB/s for random-redirection. However, for utility-redirection the replica selections result in the Amazon node(s) in the USA, thus leading to a lower but consistent average throughput of 410 KB/s.

From these observations, we come to the following decisive conclusions. Although utility-redirection outcomes sensible replica selection in terms of response time, it may not provide a high throughput performance to end-users. Nevertheless, being focused on maximizing the utility of the MetaCDN system; it results in high utility for content delivery. We provide sufficient results to support this claim in the next section.

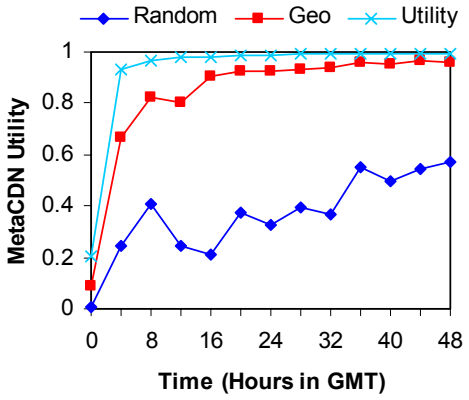


Figure 7.6: MetaCDN utility over time.

7.7.3 MetaCDN Utility

Figure 7.6 shows how MetaCDN utility is varied during the testbed experiment upon replica selection for incoming content requests. Here we have used the utility values averaged over three deployed MetaCDN gateways in Asia/Australia, Europe and North America. We observe that utility-redirection produces the highest utility in the system by selecting the most active replicas to serve users. It also improves the traffic activities and contributes to uplifting MetaCDN's content-serving ability. It should be noted that there is a warm-up phase at the beginning of the 48 hours experiment during which the replicas are populated with content requests, resulting in low utility values. This is visible during the initial hours for utility and geo-redirection.

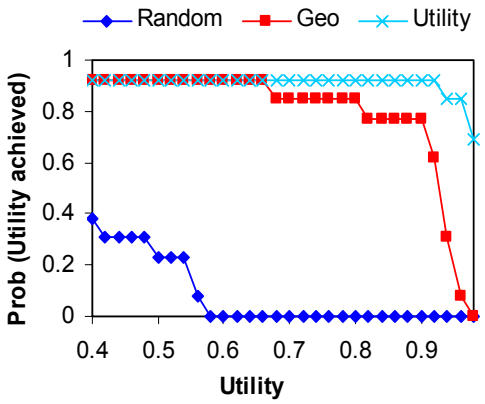


Figure 7.7: Probability of achieving specified utility.

To emphasize the content-serving ability of MetaCDN, we now present the probability that the system can achieve required minimum utility. The intention is to show to what extent the system can maximize its own profit. Figure 7.7 presents the probability (or the fraction of time) that the system observes a utility above a certain utility level during the experiment. The higher the probability, the more likely it is that the specified utility level could be achieved. From the figure, it is noticeable that utility-redirection outperforms other alternatives, as it often produces over 0.95 utility for MetaCDN with a 0.85 probability. Geo-redirection performs well as it has a 0.77 probability that it can achieve 0.9 utility. Finally, random-redirection performs the worst and it can only achieve close to 0.56 utility for MetaCDN with a probability of 0.23. Therefore, a MetaCDN administrator may utilize a request-redirection policy apart from random, in order to maximize the system's content-serving ability.

7.7.4 Content Provider's Perceived Utility

We now shed light on the content provider's perceived utility (surplus) or benefits (Section 7.4.2) from using MetaCDN, as reported in Table 7.3. For this purpose, we consider a scenario with 8 client locations belonging to 8 content providers. We use weighted throughput and normalized values of perceived response times from the client locations of respective content providers using the utility-redirection policy. We also measure the direct replica access time from each of the client locations. Finally, we make use of pricing information of Amazon S3, as reported by Broberg et al. [29]. The perceived utilities stated in Table 7.3 are to be considered representative, as

they may vary depending on the heterogeneous pricing structure of different storage cloud providers. In addition, different request-redirection policy leads to different perceived utility for a content provider.

Table 7.3: Content providers' benefits on user perceived performance.

Content Provider	End-user Location	Average Response Time (s)	Average Direct Replica Access Time (s)	Throughput	Perceived Utility (%)
1	Paris	0.99	0.747	475.39 KB/s	13.31
2	Innsbruck	1.03	0.955	518.67 KB/s	29.29
3	Poznan	1.52	0.667	1.80 MB/s	68.99
4	Beijing	4.17	1.337	176.54 KB/s	58.14
5	Melbourne	1.72	0.75	413.15 KB/s	25.32
6	Atlanta	1.09	0.605	3.35 MB/s	66.47
7	Irvine	1.06	0.391	504.74 KB/s	25.38
8	Rio de Janeiro	1.81	1.17	1.14 MB/s	34.02

We observe that a content provider's perceived utility is heavily dependent on the throughput that its end-users receive. Therefore, content providers whose end-users benefit from high throughput also realize high quantitative benefit from using the MetaCDN system. As the clients in Poznan and Atlanta experience highest throughputs, the content provider's surplus for this locations are also highest. Using utility-redirection, the average throughput experienced in Paris is the lowest, thus leading to the least content provider's surplus. Nevertheless, the utility-based request-redirection policy, being antagonistic to content provider's utility, can still assists in resulting reasonable perceived utility for the content providers. Thus, the above results show that the MetaCDN system is helpful for content providers, even at times when it uses a request-redirection technique to maximize its own utility.

7.8 Summary and Conclusion

MetaCDN, a type of Content Delivery Cloud, provides a cost-effective solution for responsive, scalable, and transparent content delivery services by harnessing the resources of multiple storage cloud providers. It provides sensible performance and availability benefits without requiring the content providers to build or manage complex content delivery infrastructure themselves. In this chapter, we have presented an approach to maximize the utility for content delivery via MetaCDN,

achieved by implementing a utility-aware request-redirection policy in the system. The utility metric reflects the traffic activities in the MetaCDN overlay system and exhibit the usefulness of its replica infrastructure. We have used the measured utility to devise the implemented request-redirection policy and quantify the benefits of a content provider for using MetaCDN. We have conducted *proof-of-concept* experiments on a global testbed to evaluate the performance of our approach. From the results obtained, we conclude that the utility of MetaCDN is maximized by using utility-based request-redirection to provide sensible replica selection and consistent average response time; however, with the cost of lower throughput in comparison to other candidate request-redirection policies. In contrast, a content provider's benefit is enhanced with improvement of the perceived throughput through MetaCDN. Therefore, a MetaCDN administrator should use a redirection policy based on the objective of either maximizing system utility or a content provider's utility.

We aim at conducting a set of future developments in MetaCDN, which are outside the scope of this thesis. Our list of future investigations includes the development of advanced request-redirection techniques and pricing policies to benefit both the MetaCDN overlay system and content providers; and on-demand autonomic management (expansion/contraction) of replica and gateway deployment. In the next chapter, we conclude the thesis with the elaboration of the future research directions.

Chapter 8

Conclusion and Future Directions

The purpose of this thesis is to develop solutions to enable internetworking of multiple content delivery services to ensure Quality of Service (QoS)-constrained content delivery, by combating against resource over-provisioning, Service Level Agreement (SLA) violation, and excessive operational cost for a CDN provider. Throughout the thesis, we introduce landmark achievements towards realizing this aim. In this chapter, we first summarize the contributions and findings to perform a reality check for the thesis objectives in order to ascertain whether our achievements are in line with them. Then we lay out a list of research issues for future investigations.

8.1 Summary

Present trends in content networks and content networking capabilities give rise to the interest in interconnecting CDNs, thus allowing providers to revel on increased scale and reach than that is achievable individually. Real world case studies show the significance of such cooperation. In this thesis, the technology for interconnection and interoperation between CDNs is termed as "peering" and we call the resultant system as "CDN peering". In this context, we have identified the fundamental research issues and challenges to address the core problems of load index measurement and dissemination, resource discovery, request-redirection, load distribution, and network utility measurement. We set forth our goals to address these key issues to achieve significant gains in cost effectiveness, performance, scalability,

and coverage through the CDN peering framework. In addition to highlighting the implications of our work in research domain, we also target demonstrating its applicability in the practical context through a *proof-of-concept* implementation.

This thesis investigates the state of the art in CDNs in terms of common trends, solutions, applications, features and implementation techniques by capturing a snapshot of the current CDN landscape. Our investigation has revealed the lack of a taxonomy to perform a complete categorization of CDNs in terms of functional and non-functional attributes. To address this need, a comprehensive taxonomy is developed to categorize the related solutions and systems. The taxonomy is also mapped to representative CDNs to demonstrate its applicability. A glimpse of the CDN technologies covered in our study and the developed taxonomy is helpful to analyze this thriving field. This investigation also serves as the basis for our innovations related to the internetworking of multi-provider content delivery services, realized through CDN peering.

Our analysis of the research initiatives in relation to CDN peering unveils only a modest progress on the framework, policies and enabling technologies to support peering. Moreover, there are a number of technical and non-technical challenges, ranging from the technological complexity to the legal and commercial operational issues, that may hinder the growth of CDN peering. These challenges can be addressed by either following an integrated overlay or a brokering approach. These lessons led to the development of an architecture, which follows an overlay approach, to assist the formation of CDN peering. The proposed architecture endeavors to attain economics of scale, extended scalability and resource sharing with other CDNs. Based on this architecture, CDN peering can be short-term wherein CDNs operate to handle flash crowds, or long-term in which they explore the delivery of specialized services. The description of SLA negotiation and policy management in this context ascertain the feasibility and position the proposed architecture as a policy-driven framework to enable peering. In addition, the introduction of two complementary models to assist CDN peering, following a brokering approach, provides us flexibility to follow alternative approaches. Notably, one of these models has been used to

develop the MetaCDN system [29, 157] by harnessing resources from multiple storage cloud providers to ensure high performance content delivery. Thus, the applicability of this thesis in practical context is validated and our reasoning to propose alternative models to enable peering is justified.

Analytical modeling is helpful to represent a system's behavioral performance by characterizing its key features. This thesis utilizes the fundamentals of queuing theory to develop QoS-oriented models that provide a foundation for performing effective peering between CDNs. The delineated claims are backed up with sufficient results to demonstrate that CDN peering achieves QoS in service delivery to end-users. The model-based approach seeks to achieve scalability for a CDN in a user transparent manner. With the findings based on these models, this thesis opens up to devise enabling technologies for CDN peering. Furthermore, it leads towards a simulation-based advanced system analysis for the developed mechanisms.

To ensure the success of peering and the effectiveness of operations for content delivery in a CDN peering system, this thesis presents novel schemes to perform resource discovery, server selection, and request-redirection to handle high load skews, such as during flash crowds. These techniques are interleaved to collectively perform load sharing in the CDN peering system. The proposed resource discovery algorithm, reminiscent of the publish/subscribe paradigm, strives for scalability and system decoupling via an offline approach. In addition, this thesis introduces a dynamic request-redirection technique that reacts to overloaded server conditions by steering excessive end-user requests to the least loaded servers, so as to minimize redirection cost in terms of traffic load and network proximity. Extensive simulation analysis, considering practical constraints and critical system parameters, reveals the novelty of the proposed techniques. This thesis positions the proposed request-redirection technique to be used as a complementary scheme to the commonly used DNS-based request-routing. Through the presented strategies this thesis also seeks to overcome the shortcomings of DNS-dispatching policy by ensuring fine-grained optimal server selections even under high load situations.

The notion of utility is quite enticing in relation to content delivery. In our context,

utility refers to the quantification of traffic activities in the distributed network of the CDN peering system. While only few previous research [39, 92, 141, 204, 207] have considered utility in their work, this thesis sets out to introduce a utility model that provides a customer view of the CDN peering system health for different traffic types. The utility model has been evaluated through simulations, which reveals many interesting observations on the impact of critical system parameters on the perceived utility. Experiment results demonstrate that the system's utility is essentially tailored to the participating CDN providers' service capacity, geographical proximity, network conditions and incoming request traffic. With this study, this thesis takes a further step forward to provide incentives for a better system design. Furthermore, this utility analysis serves as the building block for the development of a utility-based request-redirection strategy and a *proof-of-concept* implementation in the MetaCDN [29, 157] system.

This thesis provides an implementation perspective by presenting a measurement study on the global MetaCDN testbed spanning six continents. As a type of Content Delivery Cloud [60, 154], MetaCDN embraces the mainstream adoption of Cloud Computing [35] to build a poor man's CDN. While MetaCDN ensures high performance content delivery services to Internet end-users, its ultimate goal is to improve its own utility, in terms of network traffic, and uplift the usefulness of its replica infrastructure. With this goal, this thesis describes the implementation of the previously introduced utility model to plug in a utility-based request-redirection technique within MetaCDN. This thesis then demonstrates empirical results based on an evaluation study that has been conducted over a period of 48 hours. Performance results reveal that with the proposed redirection scheme, MetaCDN ensures sensible performance in terms of response time and throughput, while resulting in high system utility. Notably, it also assists in perceiving reasonable utility for the content providers, thus exhibiting MetaCDN's benefit for content providers.

To summarize, this thesis has laid the foundation for CDN peering with a novel suite of architectural models, innovative techniques, and practical tools that are efficient for leveraging and organizing multi-provider resources, delivering content,

characterizing system performance, sharing loads, and improving network utility. With these novel contributions, this thesis opens up avenues for future research in relation to multi-provider content delivery services and Content Delivery Clouds.

8.2 Future Directions

A number of future research directions in relation to this thesis can be devised. In this section, we populate an indicative list, realizing the awaited technological innovations in this area in the coming years. This listing also includes some of the research issues related to CDN peering that have been mentioned in Chapter 3 (Section 3.6); however, have not been explicitly addressed in this thesis. While elaborating on the future research topics, we provide pointers to existing literature so as to lay out a comprehensive research roadmap to the CDN community.

8.2.1 Content Replication in Cooperative Domain

The issue of effective replication and caching of content is significant for CDN peering. The concept of caching "hot" content is not new, but in the context of cooperative content delivery, there will be significant competing considerations. We foresee future research to lead to the outcome of dynamic, scalable, and efficient replication mechanisms that cache content on demand with respect to the locality of requests, focusing on regions where specific content is needed most. Innovative solutions integrating replication and caching is expected in the management of dynamic and personalized content in the cooperative domain. In this context, cooperative caching techniques [84, 143, 168] can be useful. To further improve CDN performance, adaptive replication techniques can be developed that will involve distributed caching [25], inter-CDN surrogate clustering [82], and content clustering [52].

For content management based on user preferences, a comprehensive model is required for the CDN peering system. Alike Web personalization [137], user preferences can be automatically learned from content request and usage data by using data mining techniques [152]. Data mining over the CDN peering system can exploit significant performance improvement through dealing with proper management of traffic, pricing and accounting/billing in the cooperative domain. Researchers can

refer to previous work [49, 51, 52, 108] and the references therein for more information on innovative content replication techniques. Detailed information on the integrated use of caching and replication as well as cache consistency mechanisms is available in existing literature [38, 201-203].

8.2.2 Active Content

The rapid growth of Peer-to-Peer (P2P) techniques and the improvement of digital content production and retrieval tools may lead to the proliferation of user-generated active content. This type of content will be generated with a great deal of autonomy to give it the self-awareness feature. Subject to autonomic replication, it can be wrapped with an incentive-based lightweight cooperative agent with forecasting logic to predict end-user demand. Active content assures to retain its face-value, however, can have different context-dependent meaning and interpretations. It can be represented with latent meta-data where context-oriented properties can be added or removed during replication. However, it is necessary to maintain the provenience information along with the production and transformation of active content so that the master copy can always be retrieved from the origin. Research in this context may draw inspiration from several technological paradigms such as Web crawling, agent systems, economic principals, and social science. For this purpose, research related to autonomic content delivery, such as autonomic replication of Web applications [195] and policy-based content delivery in an active network [132] can be useful.

8.2.3 On the Economics of Cooperation

In the thesis we mostly focus on developing efficient and scalable enabling technologies for CDN peering. What we have not directly addressed, however, is the mechanism to incentivize peering to keep the providers motivated to contribute resources. To ensure sustained resource sharing, sufficient incentives should be provided to all participating entities. Use of economic models in this context can represent a dynamic scenario and make the system more manageable through regulating and analyzing the emergent marketplace behavior. Our initial work in this context has been reported previously [158], and is briefly covered in Chapter 3.

In the market models for CDN peering, a CDN is considered as an independent economic agent for buying and selling content. These models are useful to emphasize the QoS-oriented aspects of peer selection and result in tractable analytical solutions of the underlying games. They are also used to analyze the sensitivity of different performance metrics such as the cost, net benefit, value and popularity of the content, and transport cost. In this regard, economic models [54-56, 74, 113] and CDN pricing models [13, 86, 99] can be referred to develop advanced pricing models.

8.2.4 Scalability of Dynamic Content Delivery

The CDN peering system can be coupled with the capability to deliver dynamic content in a scalable manner. Dynamic content, e.g. scripts, animations, DHTML or XML pages, are generated on-demand using Web applications based on user requests. Such content generation is customized depending on a given user profile and characteristics, with the use of scalable Web application hosting techniques such as edge computing [178], context-aware data caching [32, 197], data replication [197], and content blind data caching [197]. Instead of replicating the dynamic pages generated by a Web server, these techniques aim at replicating the means of generating content over multiple edge servers. Existing literature also includes a suite of algorithms for jointly optimizing the performance of dynamic content delivery by reducing end-user response time, while minimizing the resource utilization [183]. Industry efforts such as EdgeSuite content distribution from Akamai and IBM WebSphere edge services have emerged to provide usage-based application and dynamic content delivery. However, the effectiveness of the proposed solutions is still highly dependent on the access patterns of applications [38]. The risk of creating bottleneck in the back-end layer of the Web system is still one of the main issues that hinder the scalability of dynamic content delivery. Moreover, the shifting of Web 2.0 [146] towards personalized content, the presence of user profile information, and the convergence of Web 2.0 with user mobility would disrupt content delivery services due to user migration among edge nodes. Therefore, research initiatives are to be directed to find solutions addressing these issues.

8.2.5 Integration of Media Streaming

Hosting of on-demand media streaming service is challenging because of the enormous network bandwidth required to simultaneously deliver large amount of content to end-users. To avoid network congestions and to improve performance, P2P techniques can be used to extend the CDN peering system as an adaptive CDN. In such a system, content storage and workload from streaming server, network and storage resources are offloaded to the end-users workstations. The fundamental idea is to allow multiple subscribing peers to serve streams of the same video content simultaneously to a consuming peer rather than the traditional single server-to-client streaming model, while allowing each peer to store only a small portion of the content. In this regard, prior solutions such as cost-effective media streaming [221] and adaptive streaming CDN [83, 85] can be useful.

8.2.6 Mobility for Content Delivery

Mobile networks are increasingly becoming popular for distributing information to a large number of highly dynamic users. In comparison to wired networks, mobile networks are distinguished by potentially much higher variability in demand due to the user mobility. Content delivery in CDN peering can take into account this mobility notion, by potentially considering very high spatial and temporal demand variations to dynamically reconfigure the system, and to minimize the total traffic over the network backbone. A mobility model in this context can be designed to allow the access of accurate and up-to-date information and enterprise applications. We point to a few research initiatives [4, 130] regarding the models on mobile CDNs.

8.2.7 Applicability of Anycasting

Although examined early in the CDN evolution process, IP anycasting [153] was not considered as a viable approach for request-redirection in CDNs. This is due to the lack of load awareness and unwanted side effects of Internet routing changes on the IP anycasting mechanism. Nevertheless, the emergence of route control mechanisms coupled with external intelligence to allow dynamic route selections [212, 214] and a recent study on anycast-based measurement [17] shed light on the possibility to real-

ize CDN redirection using IP anycast. Lately, researchers have presented the architecture of an anycast CDN with proximal routing [8], along with a method to handle connection disruptions due to routing changes [7]. These findings can arouse interest in the CDN community to explore the usage of anycast-based redirection for CDN peering, by conducting operational experiment in real CDN deployments.

8.2.8 Energy-Aware Content Delivery

The CDN peering system characterizes a large-scale distributed system comprising geographically spanned server clusters. Such a system consumes huge amount of electricity, thus leading to high energy cost. Conventionally, the approach to reduce energy cost is to decrease the amount of the consumed energy. Novel techniques can be developed which reduce the energy cost through cost-aware routing strategies [175]. An energy-aware request-routing technique should consider end-user's geographical proximity, energy usage and cost, and incoming traffic load for directing end-users to the most cost-effective replica. By enabling energy-efficient content delivery through the CDN peering system, participating providers can reduce their collective energy cost. While energy-aware content delivery and request-redirection techniques are economically beneficial for commercial CDNs, a third-party provider (e.g. MetaCDN) may be interested in attaining social welfare by reducing the environmental impact of energy consumption. Therefore, intelligent request-redirection techniques can be developed that reduces the carbon footprint (CO_2 emission) of providers. In this context, researchers can refer to existing literature [48, 50, 80, 175].

8.2.9 Enhancement for Content Delivery Clouds

Extension of traditional CDNs model to Content Delivery Clouds enhance capabilities to deliver services that are not only limited to Web applications, but also include storage, raw computing or access to any number of specialized services. The rising of Content Delivery Clouds call for a systematic understanding and practical realization of the fundamental issues in current network management and control. In addition, it initiates potential research that focuses on identifying necessary application requirements, enhancing scalability, system robustness, usability and access performance,

low cost, data durability, and support for security and privacy.

The MetaCDN overlay system [29, 157] (Chapter 7) is a step towards providing flexible and adaptable content delivery services by harnessing the state-of-the-art availability and reliability features, and security measures of upstream storage cloud providers with little overhead due to load redirection. Future investigations on the MetaCDN system include active measurement approaches (as in the WhyHigh System [117]) for QoS-based or probabilistic request-redirection, automatic scaling of MetaCDN gateway and replica infrastructure, and dynamic pricing.

An extension to the MetaCDN QoS Monitor and its integration to the MetaCDN system are in progress to gather active measurement data from probes or agents deployed across the Internet to improve end-user performance. These agents operate at different locations and track the performance (response time, throughput, and reliability) that they experience in their locality when accessing content via MetaCDN. The recorded QoS measurement data is passed to the closest MetaCDN gateway that can make use of this information for advanced request-redirection.

Finally, a dynamic pricing policy for MetaCDN should be in place to ensure system profit as well as benefits and social welfare for the subscribing content providers and their end-users. For this purpose, dynamic pricing models for clouds [11, 174] can be useful.

References

[1] K. Aberer and M. Hauswirth, "An overview on peer-to-peer information systems," *Proc. Workshop on Distributed Data and Structures (WDAS'02)*, pp. 171-188, Carleton Scientific, 2002.

[2] S. Adler, "The Slashdot effect: An analysis of three Internet publications," *Linux Gazette,* vol. 38, 1999.

[3] D. Agrawal, J. Giles, and D. C. Verma, "On the performance of content distribution networks," *Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'01)*, The Society for Modeling and Simulation International (SCS), 2001.

[4] W. M. Aioffi, G. R. Mateus, J. M. de Almeida, and A. A. F. Loureiro, "Dynamic content distribution for mobile enterprise networks," *IEEE Journal on Selected Areas in Communications,* vol. 23, no. 10, pp. 2022-2031, 2005.

[5] Akamai Technologies, Inc., "Akamai to acquire Netli," Press Release, http://www.akamai.com/html/about/press/releases/2007/press_020507.html, 5 February, 2007.

[6] Akamai Technologies, Inc., "Akamai delivers record streaming and Web content during historic presidential inauguration," Press release, http://www.akamai.com/html/about/press/releases/2009/press_012009.html, 20 January, 2009.

[7] Z. Al-Qudah, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van der Merwe, "Anycast-aware transport for content delivery networks," *Proc. 18th International Conference on World Wide Web (WWW'09)*, pp. 301-310, ACM New York, NY, USA, 2009.

[8] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van der Merwe, "Anycast cdns revisited," *Proc. 17th International Conference on World Wide Web (WWW'08)*, pp. 277–286, ACM Press, NY, USA, 2008.

[9] L. Amini, A. Shaikh, and H. Schulzrinne, "Modeling redirection in geographically diverse server sets," *Proc. 12th International Conference on World Wide Web (WWW'03)*, pp. 472-481, ACM Press, NY, USA, 2003.

[10] L. Amini, A. Shaikh, H. Schulzrinne, I. B. M. Res, and N. Y. Hawthorne, "Effective peering for multi-provider content delivery services," *Proc. IEEE INFOCOM'04*, pp. 850-861, IEEE CS Press, Los Alamitos, CA, USA, 2004.

[11] A. Anandasivam and M. Premm, "Bid price control and dynamic pricing in clouds," *Proc. 17th European Conference on Information Systems (ECIS'09)*, 2009.

[12] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content

distribution technologies," *ACM Computing Surveys,* vol. 36, no. 4, pp. 335-371, 2004.

[13] C. Aperjis, M. J. Freedman, and R. Johari, "Peer-assisted content distribution with prices," *Proc. 2008 ACM CoNEXT Conference*, ACM Press, NY, USA, 2008.

[14] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup Web site," *IEEE network,* vol. 14, no. 3, pp. 30-37, 2000.

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and I. Stoica, *Above the clouds: A Berkeley view of cloud computing,* Technical Report, UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[16] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," *Communications of the ACM,* vol. 46, no. 2, pp. 43-48, 2003.

[17] H. Ballani, P. Francis, and S. Ratnasamy, "A measurement-based deployment proposal for IP anycast," *Proc. ACM SIGCOMM Conference on Internet Measurement*, pp. 231-244, ACM Press, NY, USA, 2006.

[18] A. Barbir, O. Batuner, A. Beck, T. Chan, and H. Orman, "Policy, authorization, and enforcement requirements of the open pluggable edge services (OPES)," *Internet Engineering Task Force*, http://www.ietf.org/rfc/rfc3838.txt, 2004.

[19] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known content network (CN) request-routing mechanisms," *Internet Engineering Task Force RFC 3568*, http://www.ietf.org/rfc/rfc3568.txt, 2003.

[20] A. Barbir, R. Chen, M. Hofmann, H. Orman, R. Penno, and G. Tomlinson, "An architecture for open pluggable edge services (OPES)," *Internet Engineering Task Force RFC 3835*, http://www.ietf.org/rfc/rfc3835.txt, 2004.

[21] P. Barford and M. Crovella, "A performance evaluation of hyper text transfer protocols," *ACM SIGMETRICS Performance Evaluation Review,* vol. 27, no. 1, pp. 188-197, 1999.

[22] N. Bartolini, E. Casalicchio, and S. Tucci, "A walk through content delivery networks," *Lecture Notes in Computer Science, Performance Tools and Applications to Networked Systems: Revised Tutorial Lectures of IEEE Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'03),* vol. 2965, pp. 1-25, 2004.

[23] R. J. Bayardo Jr, R. Agrawal, D. Gruhl, and A. Somani, "YouServ: A web-hosting and content sharing tool for the masses," *Proc. 11th International World Wide Web Conference (WWW'02)*, pp. 345-354, ACM Press, NY, USA, 2002.

[24] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spatscheck, and W. Sturm, "CDN brokering," *Computer Communications,* vol. 25, no. 4, pp. 393-402, 2002.

[25] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," *Proc. IEEE INFOCOM'10*, IEEE CS Press, Los Alamitos, CA, USA, 2010.

[26] J. Bouman, J. Trienekens, and M. Van der Zwan, "Specification of service level agreements, clarifying concepts on the basis of practical research," *Proc. Software Technology and Engineering Practice Conference*, IEEE CS Press, Los Alamitos, CA, USA, 1999.

[27] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," *Proc. IEEE INFOCOM'05*, IEEE CS Press, Los Alamitos, CA, USA, 2005.

[28] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *IEEE Computer,* vol. 33, no. 5, pp. 59-67, 2000.

[29] J. Broberg, R. Buyya, and Z. Tari, "MetaCDN: Harnessing 'Storage Clouds' for high performance content delivery," *Journal of Network and Computer Applications,* vol. 32, no. 5, pp. 1012-1022, 2009.

[30] J. Broberg, P. Zeephongsekul, and Z. Tari, "Approximating bounded general service distributions," *Proc. IEEE Symposium on Computers and Communications (ISCC'07)*, pp. 817-823, IEEE CS Press, Los Alamitos, CA, USA, 2007.

[31] K. Bubendorfer and W. Thomson, "Resource management using untrusted auctioneers in a Grid economy," *Proc. 2nd IEEE International Conference on eScience and Grid Computing (eScience'06)*, IEEE CS Press, Los Alamitos, CA, USA, 2006.

[32] T. Buchholz, I. Hochstatter, and C. Linnhoff-Popien, "A profit maximizing distribution strategy for context-aware services," *Proc. 2nd International Workshop on Mobile Commerce and Services (WMCS'05)*, pp. 144-153, IEEE CS Press, Los Alamitos, CA, USA, 2005.

[33] R. Buyya, A.-M. K. Pathan, and A. Vakali (Eds.), *Content Delivery Networks*: Springer, Germany, 2008.

[34] R. Buyya, A. M. K. Pathan, J. Broberg, and Z. Tari, "A case for peering of content delivery networks," *IEEE Distributed Systems Online,* vol. 7, no. 10, pp. 3, 2006.

[35] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems,* vol. 25, no. 6, pp. 599-616, 2009.

[36] S. Calo, D. Verma, J. Giles, and D. Agrawal, "On the effectiveness on content distribution networks," *Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'02)*, The Society for Modeling and Simulation International (SCS), 2002.

[37] C. W. Cameron, S. H. Low, and D. X. Wei, "High-density model for server allocation and placement," *Proc. ACM SIGMETRICS'02*, pp. 152-159, ACM Press, NY, USA, 2002.

[38] C. Canali, V. Cardellini, M. Colajanni, and R. Lancellotti, "Content delivery and management," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 105-126: Springer-Verlag, Germany, 2008.

[39] C. Canali, M. Rabinovich, and Z. Xiao, "Utility computing for Internet applications," *Web Content Delivery*, X. Tang, J. Xu and S. T. Chanson, eds., pp. 131–151: Springer, 2004.

[40] P. Cao, J. Zhang, and K. Beach, "Active cache: Caching dynamic contents on the Web," *Proc. IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, pp. 373-388, Springer, 1998.

[41] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The state of the art in locally distributed Web-server systems," *ACM Computing Surveys,* vol. 34, no. 2, pp. 263-311, 2002.

[42] V. Cardellini, M. Colajanni, and P. Yu, "Redirection algorithms for load sharing in distributed Web-server systems," *Proc. 19th IEEE International Conference on Distributed Computing Systems (ICDCS'99)*, pp. 528-535, IEEE CS Press, Los Alamitos, CA, USA, 1999.

[43] V. Cardellini, M. Colajanni, and P. S. Yu, "Efficient state estimators for load control policies in scalable Web server clusters," *Proc. 22nd Annual International Computer Software and Applications Conference (COMPSAC'98)*, pp. 449-457, IEEE CS Press, Los Alamitos, CA, USA, 1998.

[44] V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic load balancing on Web-server systems," *IEEE Internet Computing,* vol. 3, no. 3, pp. 28-39, 1999.

[45] V. Cardellini, M. Colajanni, and P. S. Yu, "Geographic load balancing for scalable distributed Web systems," *Proc. International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'00)*, IEEE CS Press, Los Alamitos, CA, USA, 2000.

[46] V. Cardellini, M. Colajanni, and P. S. Yu, "Request redirection algorithms for distributed Web systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 14, no. 4, pp. 355-368, 2003.

[47] S. Ceri and G. Pelagatti, *Distributed Databases Principles and Systems*: McGraw-Hill, NY, 1984.

[48] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," *Proc. IEEE INFOCOM'08*, IEEE CS Press, Los Alamitos, CA, USA, 2008.

[49] C. M. Chen, Y. Ling, M. Pang, W. Chen, S. Cai, Y. Suwa, and O. Altintas, "Scalable request routing with next-neighbor load sharing in multi-server environments," *Proc. 19th International Conference on Advanced Information*

Networking and Applications, pp. 441-446, IEEE CS Press, Los Alamitos, CA, USA, 2005.

[50] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," *Proc. USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'08)*, USENIX Association Berkeley, CA, USA, 2008.

[51] Y. Chen, R. H. Katz, and J. Kubiatowicz, "Dynamic replica placement for scalable content delivery," *Lecture Notes in Computer Science, Peer-to-Peer Systems: Revised Papers of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02),* vol. 2429, pp. 306–318, 2002.

[52] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and adaptive Web replication using content clustering," *IEEE Journal on Selected Areas in Communications,* vol. 21, no. 6, pp. 979-994, 2003.

[53] G. L. Choudhury, K. K. Leung, and W. Whitt, "Efficiently providing multiple grades of service with protection against overloads in shared resources," *AT&T Technical Journal,* vol. 74, no. 4, pp. 50-63, 1995.

[54] N. Christin and J. Chuang, "On the cost of participating in a peer-to-peer network," *Lecture Notes in Computer Science, Peer-to-Peer Systems III: Revised paper of 4th International Workshop on Peer-to-Peer Systems (IPTPS'04),* vol. 3279, pp. 22-32, 2004.

[55] N. Christin and J. Chuang, "A cost-based analysis of overlay routing geometrics," *Proc. IEEE INFOCOM'05,* vol. 4, pp. 2566-2577, IEEE CS Press, Los Alamitos, CA, USA, 2005.

[56] N. Christin, J. Chuang, and J. Grossklags, "Economics-informed design of CDNs," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 183-210: Springer-Verlag, Germany, 2008.

[57] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review,* vol. 33, no. 3, pp. 3-12, 2003.

[58] B. Ciciani, F. Calderoni, A. Santoro, and F. Quaglia, "Modeling of QoS-oriented content delivery networks," *Proc. 13th IEEE Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'05)*, pp. 341-344, IEEE Press, NJ, USA, 2005.

[59] A. Cobham, "Priority assignment in waiting line problems," *Journal of the Operations Research Society of America,* vol. 2, no. 1, pp. 70-76, 1954.

[60] R. Cohen, "Content delivery cloud (CDC)," *ElasticVapor: Life in the Cloud*, http://www.elasticvapor.com/2008/10/cloud-content-delivery-cd.html, 2008.

[61] M. Colajanni and P. S. Yu, "A performance study of robust load sharing strategies for distributed heterogeneous Web server systems," *IEEE Transac-*

tions on Knowledge and Data Engineering, pp. 398-414, 2002.

[62] M. Colajanni, P. S. Yu, and V. Cardellini, "Dynamic load balancing in geographically distributed heterogeneous Web servers," *Proc. 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pp. 295-302, IEEE CS Press, Los Alamitos, CA, USA, 1998.

[63] M. Colajanni, P. S. Yu, and D. M. Dias, "Analysis of task assignment policies in scalable distributed Web-server systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 9, no. 6, pp. 585-600, 1998.

[64] M. Conti, E. Gregori, and W. Lapenna, "Replicated Web services: A comparative analysis of client-based content-delivery policies," *Lecture Notes in Computer Science, Web Engineering and Peer-to-Peer Computing: Revised Papers of Networking'02 Workshops,* vol. 2376, pp. 53-68, 2002.

[65] M. Conti, E. Gregori, and F. Panzieri, "QoS-based architectures for geographically replicated Web servers," *Cluster Computing,* vol. 4, no. 2, pp. 109-120, 2001.

[66] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of scheduling*: Dover Publications, 2003.

[67] I. Cooper, I. Melve, and G. Tomlinson, "Internet Web replication and caching taxonomy," *Internet Engineering Task Force RFC 3040*, http://www.ietf.org/rfc/rfc3040.txt, 2001.

[68] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: evidence and possiblecauses," *IEEE/ACM Transactions on Networking,* vol. 5, no. 6, pp. 835-846, 1997.

[69] M. E. Crovella, M. S. Taqqu, and A. Bestavros, "Heavy-tailed probability distributions in the World Wide Web," *A practical guide to heavy tails: statistical techniques and applications*, R. J. Adler, R. E. Feldman and M. S. Taqqu, eds., pp. 3-25: Birkhauser, 1998.

[70] M. Dahlin, "Interpreting stale load information," *IEEE Transactions on Parallel and Distributed Systems,* vol. 11, no. 10, pp. 1033-1047, 1999.

[71] B. D. Davison, "Web caching and content delivery resources," http://www.web-caching.com, 2009.

[72] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A model for content internetworking (CDI)," *Internet Engineering Task Force RFC 3466*, http://www.ietf.org/rfc/rfc3466.txt, 2003.

[73] V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian, "A quality of service management framework based on user expectations," *Lecture Notes in Computer Science, Proc. 1st International Conference on Service-Oriented Computing (ICSOC'03),* vol. 2910, pp. 104-114, 2003.

[74] A. Di Stefano and C. Santoro, "An economic model for resource management

in a Grid-based content distribution network," *Future Generation Computer Systems,* vol. 24, no. 3, pp. 202-212, 2008.

[75] M. Dikaiakos and A. Stassopoulou, "Content-selection strategies for the periodic prefetching of WWW resources via satellite," *Computer Communications,* vol. 24, no. 1, pp. 93-104, 2001.

[76] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Computing,* vol. 6, no. 5, pp. 50-58, 2002.

[77] F. Douglis and M. F. Kaashoek, "Scalable internet services," *IEEE Internet Computing,* vol. 5, no. 4, pp. 36-37, 2001.

[78] J. Elson and J. Howell, "Handling flash crowds from your garage," *Proc. USENIX 2008 Annual Technical Conference (USENIX'08)*, pp. 171-184, USENIX Association Berkeley, CA, USA, 2008.

[79] O. Erçetin and L. Tassiulas, *Request routing in content distribution networks,* Technical Report, Sabanci University, Turkey, http://digital.sabanciuniv.edu/elitfulltext/3011800000049.pdf, 2003.

[80] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *Proc. 34th Annual International Symposium on Computer Architecture*, pp. 13-23, ACM Press, NY, USA, 2007.

[81] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Transactions on Networking,* vol. 9, no. 4, pp. 392-403, 2001.

[82] G. Fortino, A. Garro, S. Mascillaro, W. Russo, and M. Vaccaro, "Distributed architectures for surrogate clustering in CDNs: a simulation-based analysis," *UPGRADE-CN'09, Proc. 18th IEEE International Symposium on High Performance Distributed Computing (HPDC'09) Workshops*, pp. 3-10, ACM Press, NY, USA, 2009.

[83] G. Fortino, C. Mastroianni, and W. Russo, "Collaborative media streaming services based on CDNs," *Content Delivery Networks,* R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 297-316: Springer-Verlag, Germany, 2008.

[84] G. Fortino and W. Russo, "Using P2P, GRID and Agent technologies for the development of content distribution networks," *Future Generation Computer Systems,* vol. 24, no. 3, pp. 180-190, 2008.

[85] G. Fortino, W. Russo, C. Mastroianni, C. E. Palau, and M. Esteve, "CDN-supported collaborative media streaming control," *IEEE Multimedia,* vol. 14, no. 2, pp. 60-71, 2007.

[86] M. J. Freedman, C. Aperjis, and R. Johari, "Prices are right: Managing resources and incentives in peer-assisted content distribution," *Proc. 7th Inernational Workshop on Peer-to-Peer Systems (IPTPS'08)*, 2008.

[87] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing content

publication with Coral," *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'04)*, pp. 239-252, USENIX Association Berkeley, CA, USA, 2004.

[88] S. Gadde, J. Chase, and M. Rabinovich, "Web caching and content distribution: A view from the interior," *Computer Communications,* vol. 24, no. 2, pp. 222-231, 2001.

[89] L. Gannes, "The Obama inauguration live stream stats," http://newteevee.com/2009/01/20/the-obama-inauguration-live-stream-stats, 20 January, 2009.

[90] L. Gao, M. Dahlin, A. Nayate, J. Zheng, and A. Iyengar, "Application specific data replication for edge services," *Proc. 12th International World-Wide Web Conference (WWW'03)*, pp. 449-460, ACM Press, NY, USA, 2003.

[91] A. Garg and A. L. N. Reddy, "Mitigating denial of service attacks using QoS regulation," *Proc. International Workshop on Quality of Service (IWQoS'02)*, IEEE Communications Society, 2002.

[92] P. Gayek, R. Nesbitt, H. Pearthree, A. Shaikh, and B. Snitzer, "A Web content serving utility," *IBM Systems Journal,* vol. 43, no. 1, pp. 43-63, 2004.

[93] X. Geng, R. D. Gopal, R. Ramesh, and A. B. Whinston, "Scaling Web services with capacity provision networks," *IEEE Computer,* vol. 36, no. 11, pp. 64-72, 2003.

[94] L. Golubchik and J. C. S. Lui, "Bounding of performance measures for threshold-based queuing systems: Theory and application to dynamic resource management in video-on-demand servers," *IEEE Transactions on Computers,* vol. 51, no. 4, pp. 353-372, 2002.

[95] D. Gottfrid, "Self-service, prorated super computing fun!," *The New York Times*, http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/, 2007.

[96] L. Guo, S. Chen, Z. Xiao, and X. Zhang, "Analysis of multimedia workloads with implications for internet streaming," *Proc. 14th international Conference on World Wide Web (WWW'05)*, pp. 519-528, ACM Press, NY, USA, 2005.

[97] M. Harchol-Balter, T. Leighton, and D. Lewin, "Resource discovery in distributed networks," *Proc. 18th ACM Symposium on Principles of Distributed Computing*, pp. 229-237, ACM Press, NY, USA, 1999.

[98] M. Hofmann and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*: Morgan Kaufmann Publishers, 2005.

[99] K. Hosanagar, J. Chuang, R. Krishnan, and M. D. Smith, "Service adoption and pricing of Content Delivery Network (CDN) services," *Management Science,* vol. 54, no. 9, pp. 1579-1593, 2008.

[100] ISO, "Open Systems Interconnection, Basic Reference Model," ISO 7498,

http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256, 1989.

[101] A. K. Iyengar, M. S. Squillante, and L. Zhang, "Analysis and characterization of large scale Web server access patterns and performance," *World Wide Web,* vol. 2, no. 1, pp. 85-100, 1999.

[102] P. Jalote, *Fault Tolerance in Distributed Systems*: Prentice Hall Englewood Cliffs, NJ, 1994.

[103] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering," *Proc. Workshop on the Economics of Networks, Systems, and Computation (NetEcon'08)*, pp. 7-12, ACM Press, NY, USA, 2008.

[104] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, "The measured performance of content distribution networks," *Computer Communications,* vol. 24, no. 2, pp. 202-206, 2001.

[105] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and Web sites," *Proc. 11th International World Wide Web Conference (WWW'02)*, pp. 293-304, ACM Press, NY, USA, 2002.

[106] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," *Proc. IEEE Symposium on Security and Privacy*, pp. 211-225, IEEE CS Press, Los Alamitos, CA, USA, 2004.

[107] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds," *Proc. 2nd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'05)*, USENIX Association Berkeley, CA, USA, 2005.

[108] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Computer Communications,* vol. 25, no. 4, pp. 376-383, 2002.

[109] J. Kangasharju, K. W. Ross, and J. W. Roberts, "Performance evaluation of redirection schemes in content distribution networks," *Computer Communications,* vol. 24, no. 2, pp. 207-214, 2001.

[110] M. Karaul, Y. A. Korilis, and A. Orda, "A market-based architecture for management of geographically dispersed, replicated Web servers," *Decision Support Systems,* vol. 28, no. 1-2, pp. 191-204, 2000.

[111] F. Kargl, J. Maier, and M. Weber, "Protecting Web servers from distributed denial of service attacks," *Proc. International World Wide Web Conference (WWW'01)*, pp. 514-524, ACM Press, NY, USA, 2001.

[112] J. Kaufman, "Blocking in a shared resource environment," *IEEE Transactions on Communications,* vol. 29, no. 10, pp. 1474-1481, 1981.

[113] C. Kaya, K. Dogan, and V. Mookerjee, "An economic and operational analysis

of the market for content distribution services," *Proc. International Conference on Information Systems (ICIS'03)*, 2003.

[114] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "Packetscore: A statistics-based packet filtering scheme against distributed denial-of-service attacks," *IEEE Transactions on Dependable and Secure Computing,* vol. 3, no. 2, pp. 141-155, 2006.

[115] L. Kleinrock, *Queuing Systems-Vol. II: Computer Applications*: John Wiley & Sons, 1976.

[116] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," *Proc. 1st International Internet Measurement Workshop*, pp. 169-182, ACM Press, NY, USA, 2001.

[117] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize CDN performance," *Proc. ACM SIGCOMM Internet Measurement Conference (IMC'09)*, pp. 190-201, ACM Press, NY, USA, 2009.

[118] H. T. Kung and C. H. Wu, "Content networks: Taxonomy and new approaches," *The Internet as a Large-Scale Complex System*, K. Park and W. Willinger, eds.: Oxford University Press, 2002.

[119] M. Kwon and S. Fahmy, "Synergy: An overlay internetworking architecture," *Proc. 14th International Conference on Computer Communications and Networks (ICCCN'05)*, pp. 401-406, IEEE Press, NJ, USA, 2005.

[120] A. Lamnitchi, I. Foster, and D. C. Nurmi, "A peer-to-peer approach to resource location in grid environments," *Grid Resource Management*, J. Weglarz, J. Nabrzyski, J. Schopf and M. Stroinski, eds., pp. 413-430: Kluwer Publishing, 2003.

[121] A. M. Law and W. D. Kelton, *Simulation modelling and analysis*: MacGraw Hill, 1991.

[122] I. Lazar and W. Terrill, "Exploring content delivery networking," *IT Professional,* vol. 3, no. 4, pp. 47-49, 2001.

[123] J. Lee, "An end-user perspective on file-sharing systems," *Communications of the ACM,* vol. 46, no. 2, pp. 49-53, 2003.

[124] T. Leighton, *Akamai and cloud computing: A perspective from the edge of the cloud,* White Paper, Akamai Technologies, Inc., http://www.akamai.com/cloud, 2009.

[125] Y. Li and M. T. Liu, "Optimization of performance gain in content distribution networks with server replicas," *Proc. International Symposium on Applications and the Internet (SAINT'03)*, pp. 182–189, IEEE CS Press, Los Alamitos, CA, USA, 2003.

[126] J. Liang, R. Kumar, and K. W. Ross, "The FastTrack overlay: A measurement

study," *Computer Networks,* vol. 50, no. 6, pp. 842-858, 2006.

[127] Limelight-Networks, "Wow! 2.5 million streams for Obama inauguration," http://blog.llnw.com/2009/01/wow-25-million-streams-for-obama-inauguration, 20 January, 2009.

[128] R. Liston and E. Zegura, "Using a proxy to measure client-side web performance," *Proc. Intlernational Web Content Caching and Distribution Workshop (WCW'01)*, Elsevier, 2001.

[129] J. Lloret, M. Garcia, D. Bri, and J. R. Diaz, "Study and performance of a group-based content delivery network," *Journal of Network and Computer Applications,* vol. 32, no. 5, pp. 991-999, 2009.

[130] N. Loulloudes, G. Pallis, and M. D. Dikaiakos, "Information dissemination in mobile CDNs," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 343-366: Springer-Verlag, Germany, 2008.

[131] D. MacAskill, "Scalability: Set Amazon's servers on fire, not yours," *O'Reilly Emerging Technology Conference (ETech'07)*, http://blogs.smugmug.com/don/files/ETech-SmugMug-Amazon-2007.pdf, 2007.

[132] G. MacLarty and M. Fry, "Policy-based content delivery: An active network approach," *Computer Communications,* vol. 24, no. 2, pp. 241-248, 2001.

[133] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang, "A precise and efficient evaluation of the proximity between Web clients and their local DNS servers," *Proc. USENIX 2002 Annual Technical Conference (USENIX'02)*, pp. 229-242, USENIX Association Berkeley, CA, USA, 2002.

[134] M. Masa and E. Parravicini, "Impact of request routing algorithms on the delivery performance of content delivery networks," *Proc. International Conference on Performance Computing and Communications (IPCCC'03)*, pp. 5-12, IEEE CS Press, Los Alamitos, CA, USA, 2003.

[135] R. Miller, "Microsoft building own CDN network," *Data Center Knowledge*, http://www.datacenterknowledge.com/archives/2008/01/11/microsoft-building-own-cdn-network/, 2008.

[136] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, *Peer-to-peer computing,* Technical Report, HPL-2002-57, HP Lab, http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf, 2002.

[137] B. Mobasher, R. Cooley, and J. Srivastava, "Automatic personalization based on Web usage mining," *Communications of the ACM,* vol. 43, no. 8, pp. 142-151, 2000.

[138] B. Molina, C. E. Palau, and M. Esteve, "Modeling content delivery networks and their performance," *Computer Communications,* vol. 27, no. 15, pp. 1401-1411, 2004.

[139] B. Molina, C. E. Palau Salvador, M. Esteve Domingo, I. Alonso Peña, and V. Ruiz Extremera, "On content delivery network implementation," *Computer Communications,* vol. 29, no. 12, pp. 2396-2412, 2006.

[140] R. Moore, T. A. Prince, and M. Ellisman, "Data-intensive computing and digital libraries," *Communications of the ACM,* vol. 41, no. 11, pp. 56-62, 1998.

[141] B. Mortazavi and G. Kesidis, "Model and simulation study of a peer-to-peer game with a reputation-based incentive mechanism," *Proc. Information Theory and Applications (ITA) Workshop*, University of California, San Diego, CA, USA, 2006.

[142] T. V. Nguyen, C. T. Chou, and P. Boustead, "Provisioning content distribution networks over shared infrastructure," *Proc. 11th IEEE International Conference on Networks (ICON'03)*, pp. 119-124, IEEE Press, NJ, USA, 2003.

[143] J. Ni and D. H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Communications Magazine,* vol. 43, no. 5, pp. 98-105, 2005.

[144] E. Ogston and S. Vassiliadis, "A peer-to-peer agent auction," *Proc. 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pp. 151-159, ACM Press, NY, USA, 2002.

[145] A. Oram, *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*: O'Reilly & Associates, Inc, 2001.

[146] T. Oreilly, "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications & Strategies, International Journal of Digital Economics,* vol. 65, pp. 17-37, 2007.

[147] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*: Prentice Hall, Inc., 1999.

[148] V. N. Padmanabhan and K. Sripanidkulchai, "The case for cooperative networking," *Lecture Notes in Computer Science, Peer-to-Peer Systems: Revised Papers of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02),* vol. 2429, pp. 178-190, 2002.

[149] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," *Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02)*, pp. 177-186, ACM Press, NY, USA, 2002.

[150] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-aware request distribution in cluster-based network servers," *ACM SIGPLAN Notices,* vol. 33, no. 11, pp. 205-216, 1998.

[151] V. S. Pai, L. Wang, K. S. Park, R. Pang, and L. Peterson, "The dark side of the Web: An open proxy's view," *ACM SIGCOMM Computer Communication Review,* vol. 34, no. 1, pp. 57-62, 2004.

[152] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM,* vol. 49, no. 1, pp. 101-106, 2006.

[153] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," *Internet Engineering Task Force RFC 1546*, http://www.ietf.org/rfc/rfc1546.txt, 1993.

[154] A.-M. K. Pathan, "Content delivery networks (CDNs) research directory," http://www.cs.mu.oz.au/~apathan/CDNs.html, 2009.

[155] A.-M. K. Pathan, J. Broberg, K. Bubendorfer, K. H. Kim, and R. Buyya, "An architecture for virtual organization (VO)-based effective peering of content delivery networks," *UPGRADE-CN'07, Proc. 16th IEEE International Symposium on High Performance Distributed Computing (HPDC'07) Workshops*, pp. 29-38, ACM Press, NY, USA, 2007.

[156] A.-M. K. Pathan, J. Broberg, and R. Buyya, "Internetworking of CDNs," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 389-413: Springer-Verlag, Germany, 2008.

[157] A.-M. K. Pathan, J. Broberg, and R. Buyya, "Maximizing utility for content delivery clouds," *Lecture Notes in Computer Science, Proc. 10th International Conference on Web Information Systems Engineering (WISE'09),* vol. 5802, pp. 13-28, 2009.

[158] A.-M. K. Pathan and R. Buyya, "Economy-based content replication for peering content delivery networks," *TCSC Doctoral Symposium, Proc. 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, pp. 887-892, IEEE CS Press, Los Alamitos, CA, USA, 2007.

[159] A.-M. K. Pathan and R. Buyya, "Performance models for peering content delivery networks," *Proc. 16th IEEE International Conference on Networks (ICON'08)*, pp. 1-7, IEEE Press, NJ, USA, 2008.

[160] A.-M. K. Pathan and R. Buyya, "A taxonomy of CDNs," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 33-77: Springer-Verlag, Germany, 2008.

[161] A.-M. K. Pathan and R. Buyya, "Architecture and performance models for QoS-driven effective peering of content delivery networks," *Multiagent and Grid Systems,* vol. 5, no. 2, pp. 165-195, 2009.

[162] A.-M. K. Pathan and R. Buyya, "Resource discovery and request-redirection for dynamic load sharing in multi-provider peering content delivery networks," *Journal of Network and Computer Applications,* vol. 32, no. 5, pp. 976-990, 2009.

[163] A.-M. K. Pathan and R. Buyya, "A utility model for peering of multi-provider content delivery services," *Proc. 34th IEEE International Conference on Local Computer Networks (LCN'09)*, pp. 475-482, IEEE CS Press, Los Alamitos, CA, USA, 2009.

[164] A.-M. K. Pathan, R. Buyya, and A. Vakali, "Content delivery networks: State

of the art, insights, and imperatives," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 3-32: Springer-Verlag, Germany, 2008.

[165] A.-M. K. Pathan, C. Vecchiola, and R. Buyya, "Load and proximity aware request-redirection for dynamic load distribution in peering CDNs," *Lecture Notes in Computer Science, Proc. 16th International Conference on Cooperative Information Systems (CoopIS'08),* vol. 5331, pp. 62-81, 2008.

[166] G. Peng, *CDN: Content distribution network,* Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, USA, http://citeseer.ist.psu.edu/peng03cdn.html, 2004.

[167] G. Pierre and M. van Steen, "Globule: A platform for self-replicating Web documents," *Lecture Notes in Computer Science, Proc. 6th International Conference on Protocols for Multimedia Systems (PROMS'01),* vol. 2213, pp. 1-11, 2001.

[168] G. Pierre and M. van Steen, "Globule: A collaborative content delivery network," *IEEE Communications Magazine,* vol. 44, no. 8, pp. 127-133, 2006.

[169] J. E. Pitkow, "Summary of WWW characterizations," *World Wide Web,* vol. 2, no. 1, pp. 3-13, 1999.

[170] T. Plagemann, R. Canonico, J. Domingo-Pascual, C. Guerrero, and A. Mauthe, "Infrastructures for community networks," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 367-388: Springer-Verlag, Germany, 2008.

[171] T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, and G. Urvoy-Keller, "From content distribution networks to content networks—issues and challenges," *Computer Communications,* vol. 29, no. 5, pp. 551-562, 2006.

[172] D. K. Pradhan, *Fault-Tolerant Computer System Design*: Prentice-Hall, Inc. Englewood Cliffs, NJ, USA, 1986.

[173] F. L. Presti, N. Bartolini, and C. Petrioli, "Dynamic replica placement and user request redirection in content delivery networks," *Proc. International Conference on Communications (ICC'05)*, pp. 1495-1501, IEEE CS Press, Los Alamitos, CA, USA, 2005.

[174] T. Pueschel, A. Anandasivam, S. Buschek, and D. Neumann, "Making money with clouds: Revenue optimization through automated policy decisions," *Proc. 17th European Conference on Information Systems (ECIS'09)*, 2009.

[175] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for Internet-scale systems," *Proc. ACM SIGCOMM'09*, ACM Press, NY, USA, 2009.

[176] M. Rabinovich and O. Spatscheck, *Web Caching and Replication*: Addison Wesley, USA, 2003.

[177] M. Rabinovich, Z. Xiao, and A. Aggarwal, "Computing on the edge: A platform for replicating internet applications," *Proc. 8th International Workshop on Web Content Caching and Distribution (WCW'03)*, Kluwer, 2003.

[178] M. Rabinovich, Z. Xiao, F. Douglis, and C. Kalmanek, "Moving edge side includes to the real edge–the clients," *Proc. USENIX Symposium on Internet Technologies and Systems (USITS'97)*, USENIX Association Berkeley, CA, USA, 1997.

[179] H. Rahul, M. Kasbekar, R. Sitaraman, and A. Berger, "Towards realizing the performance and availability benefits of a global overlay network " *Proc. 7th International Conference on Passive and Active Network Measurement (PAM'06)*, Springer, 2006.

[180] H. S. Rahul, M. Kasbekar, R. K. Sitaraman, and A. W. Berger, "Performance and availability benefits of global overlay routing," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 251-272: Springer-Verlag, Germany, 2008.

[181] S. Rangarajan, S. Mukherjee, and P. Rodriguez, "A technique for user specific request redirection in a content delivery network," *Proc. 8th International Web Content Caching and Distribution Workshop (WCW'03)*, Kluwer, 2003.

[182] S. Ranjan, R. Karrer, and E. Knightly, "Wide area redirection of dynamic content by internet data centers," *Proc. IEEE INFOCOM'04*, IEEE CS Press, Los Alamitos, CA, USA, 2004.

[183] S. Ranjan and E. Knightly, "High-performance resource allocation and request redirection algorithms for Web clusters," *IEEE Transactions on Parallel and Distributed Systems,* vol. 19, no. 9, pp. 1186-1200, 2008.

[184] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "DDoS-resilient scheduling to counter application layer attacks under imperfect detection," *Proc. IEEE INFOCOM'06*, pp. 1-13, IEEE CS Press, Los Alamitos, CA, USA, 2006.

[185] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *Proc. ACM SIGCOMM'01*, pp. 161-172, ACM Press, NY, USA, 2001.

[186] D. Rayburn, "CDN research data: Market sizing and pricing trends," *Streaming Media West: The Business and Technology of Online Video*, http://www.streamingmedia.com/east/presentations/CDNSummit09-CDN Pricing.pdf, 2009.

[187] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science, Pro. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01),* vol. 2218, pp. 329-350, 2001.

[188] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," *ACM SIGOPS Operating Sys-*

tems Review, vol. 36, pp. 315-328, 2002.

[189] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys,* vol. 22, no. 4, pp. 299-319, 1990.

[190] M. F. Schwartz, A. Emtage, B. Kahle, and B. C. Neuman, "A comparison of Internet resource discovery approaches," *Computing Systems,* vol. 5, no. 4, pp. 461-493, 1992.

[191] P. Shah, J. F. Pâris, J. Morgan, J. Schettino, and C. Venkatraman, "A P2P-based architecture for secure software delivery using volunteer assistance," *Proc. International Conference on Peer-to-Peer Networks (P2P'08)*, IEEE CS Press, Los Alamitos, CA, USA, 2008.

[192] A. Shaikh, R. Tewari, M. Agrawal, I. Center, and Y. Heights, "On the effectiveness of DNS-based server selection," *Proc. IEEE INFOCOM'01*, pp. 1801-1810, IEEE CS Press, Los Alamitos, CA, USA, 2001.

[193] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical Programming,* vol. 62, no. 1, pp. 461-474, 1993.

[194] V. Shnayder, *Load and proximity aware request distribution,* Technical Report, Princeton University, http://citeseer.ist.psu.edu/636009.html, 2003.

[195] S. Sivasubramanian, G. Alonso, G. Pierre, and M. van Steen, "GlobeDB: Autonomic data replication for web applications," *Proc. 14th International World Wide Web Conference (WWW'05)*, pp. 33-42, ACM Press, NY, USA, 2005.

[196] S. Sivasubramanian, G. Pierre, and M. van Steen, "Replicating Web applications on-demand," *Proc. IEEE International Conference on Services Computing (SCC'04)*, pp. 227-236, IEEE CS Press, Los Alamitos, CA, USA, 2004.

[197] S. Sivasubramanian, G. Pierre, M. van Steen, and G. Alonso, "Analysis of caching and replication strategies for Web applications," *IEEE Internet Computing,* vol. 11, no. 1, pp. 60-66, 2007.

[198] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. Van Steen, "Replication for Web hosting systems," *ACM Computing Surveys,* vol. 36, no. 3, pp. 291-334, 2004.

[199] C. D. S. Solutions, "Gnutella protocol specification v0. 4," http//www. clip2. com/GnutellaProtocol04. pdf.

[200] T. Stading, P. Maniatis, and M. Baker, "Peer-to-peer caching schemes to address flash crowds," *Lecture Notes in Computer Science, Peer-to-Peer Systems: Revised Papers of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02),* vol. 2429, pp. 203–213, 2002.

[201] K. Stamos, G. Pallis, C. Thomos, and A. Vakali, "A similarity based approach for integrated Web caching and content replication in CDNs," *Proc. 10th In-*

ternational Database Engineering and Applications Symposium (IDEAS'06), pp. 239-242, IEEE CS Press, Los Alamitos, CA, USA, 2006.

[202] K. Stamos, G. Pallis, and A. Vakali, "Integrating caching techniques on a content distribution network," *Lecture Notes in Computer Science, Proc. 10th East European Conference on Advances in Databases and Information Systems (ADBIS'06),* vol. 4152, pp. 200-215, 2006.

[203] K. Stamos, G. Pallis, and A. Vakali, "Caching techniques on CDN simulated frameworks," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 127-153: Springer-Verlag, Germany, 2008.

[204] K. Stamos, G. Pallis, A. Vakali, and M. D. Dikaiakos, "Evaluating the utility of content delivery networks," *UPGRADE-CN'09, Proc. 18th IEEE International Symposium on High Performance Distributed Computing (HPDC'09) Workshops*, pp. 11-20, ACM Press, NY, USA, 2009.

[205] K. Stamos, G. Pallis, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos, "CDNsim: A Simulation Tool for Content Distribution Networks," *ACM Transactions on Modeling and Computer Simulation*, 2009. To appear.

[206] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *IEEE/ACM Transactions on Networking,* vol. 11, no. 1, pp. 17-32, 2001.

[207] S. R. Subramanya and B. K. Yi, "Utility model for on-demand digital content," *IEEE Computer,* vol. 38, no. 6, pp. 95-98, 2005.

[208] M. Tran and W. Tavanapong, "Peers-assisted dynamic content distribution networks," *Proc. 30th IEEE International Conference on Local Computer Networks (LCN'05)*, pp. 123-131, IEEE CS Press, Los Alamitos, CA, USA, 2005.

[209] E. Turrini, *An architecture for content distribution internetworking,* Technical Report UBLCS-2004-2, University of Bologna, Italy, 2004.

[210] E. Turrini and F. Panzieri, "Using P2P techniques for content distribution internetworking: a research proposal," *Proc. International Conference on Peer-to-Peer Computing (P2P'02)*, IEEE CS Press, Los Alamitos, CA, USA, 2002.

[211] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing,* vol. 7, no. 6, pp. 68-74, 2003.

[212] J. Van der Merwe, A. Cepleanu, K. D'Souza, B. Freeman, A. Greenberg, D. Knight, R. McMillan, D. Moloney, J. Mulligan, and H. Nguyen, "Dynamic connectivity management with an intelligent route service control point," *Proc. ACM SIGCOMM Workshop on Internet Network Management*, pp. 29-34, ACM Press, NY, USA, 2006.

[213] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," *ACM Computing Surveys,* vol. 38, no. 1, 2006.

[214] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. C. Snoeren, and J. E. van der Merwe, "Wresting control from BGP: Scalable fine-grained route control," *Proc. USENIX 2007 Annual Technical Conference (USENIX'07)*, pp. 295-308, USENIX Association Berkeley, CA, USA, 2007.

[215] D. C. Verma, *Content Distribution Networks: An Engineering Approach*: John Wiley & Sons, Inc., 2002.

[216] D. C. Verma, S. Calo, K. Amiri, I. Center, and Y. Heights, "Policy-based management of content distribution networks," *IEEE network,* vol. 16, no. 2, pp. 34-39, 2002.

[217] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *Journal of finance,* vol. 16, no. 1, pp. 8-37, 1961.

[218] D. Villela and D. Rubenstein, "A queuing analysis of server sharing collectives for content distribution," *Lecture Notes in Computer Science, Proc. International Workshop on Quality of Service (IWQoS'03),* vol. 2707, pp. 41-58, 2003.

[219] D. Villela and D. Rubenstein, "Performance analysis of server sharing collectives for content distribution," *IEEE Transactions on Parallel and Distributed Systems,* vol. 16, no. 12, pp. 1178-1189, 2005.

[220] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey Review,* vol. 22, no. 176, pp. 88–93, 1975.

[221] A. K. Wah Yim and R. Buyya, "Decentralized media streaming infrastructure (DeMSI): An adaptive and high-performance peer-to-peer content delivery network," *Journal of Systems Architecture,* vol. 52, no. 12, pp. 737-772, 2006.

[222] L. Wang, V. Pai, and L. Peterson, "The effectiveness of request redirection on CDN robustness," *ACM SIGOPS Operating Systems Review,* vol. 36, pp. 345-360, 2002.

[223] L. Wang, K. S. Park, R. Pang, V. Pai, and L. Peterson, "Reliability and security in the CoDeeN content distribution network," *Proc. USENIX 2004 Annual Technical Conference (USENIX'04)*, USENIX Association Berkeley, CA, USA, 2004.

[224] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, "Terminology for policy-based management," *Internet Engineering Task Force RFC 3198*, http://www.ietf.org/rfc/rfc3198.txt, 2001.

[225] W. Willinger and V. Paxson, "Where mathematics meets the Internet," *Notices of the American Mathematical Society,* vol. 45, no. 8, pp. 961-971, 1998.

[226] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for (P2P) applications," *Proc. ACM SIGCOMM'08*, ACM Press, NY, USA, 2008.

[227] C.-S. Yang and M.-Y. Luo, "An effective mechanism for supporting content-based routing in scalable Web server clusters," *Proc. International Workshop on Parallel Processing (ICPP'99)*, pp. 240-245, IEEE CS Press, Los Alamitos, CA, USA, 1999.

[228] N. Yoshida, "Dynamic CDN against flash crowds," *Content Delivery Networks*, R. Buyya, A.-M. K. Pathan and A. Vakali, eds., pp. 275-296: Springer-Verlag, Germany, 2008.

[229] W. Zhao and H. Schulzrinne, "DotSlash: A self-configuring and scalable rescue system for handling Web hotspots effectively," *Lecture Notes in Computer Science, Proc. 9th International Workshop on Web Content Caching and Distribution (WCW'04),* vol. 3293, pp. 1-18, 2004.

[230] Y. Zhu and Y. Hu, "Efficient, proximity-aware load balancing for DHT-based P2P systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 16, no. 4, pp. 349-361, 2005.

University Library

A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Pathan, Al-Mukaddim Khan

Title:
Utility-oriented internetworking of content delivery networks

Date:
2009

Citation:
Pathan, A. K. (2009). Utility-oriented internetworking of content delivery networks. PhD thesis, Engineering - Computer Science and Software Engineering, The University of Melbourne.

Persistent Link:
http://hdl.handle.net/11343/35288

File Description:
Utility-oriented internetworking of content delivery networks

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.