

University of Wollongong - Research Online

Thesis Collection

Title: Content distribution networks over shared infrastructure : a paradigm for future content network deployment

Author: Thanh Vinh Nguyen

Year: 2005

Repository DOI:

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material.

Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.

Research Online is the open access repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

University of Wollongong Thesis Collections

University of Wollongong Thesis Collection

University of Wollongong

Year 2005

Content distribution networks over
shared infrastrucutre: a paradigm for
future content network deployment

Thanh Vinh Nguyen
University of Wollongong

Nguyen, Thanh Vinh, Content distribution networks over shared infrastrucutre: a paradigm for future content network deployment, PhD thesis, School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, 2005. <http://ro.uow.edu.au/theses/411>

This paper is posted at Research Online.
<http://ro.uow.edu.au/theses/411>

NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Content Distribution Networks over Shared
Infrastructure
A Paradigm for Future Content Network
Deployment

A thesis submitted in fulfilment of the
requirements for the award of the degree

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

Thanh Vinh Nguyen
Bachelor of Engineering (Honours Class I)

SCHOOL OF ELECTRICAL, COMPUTER
AND TELECOMMUNICATIONS ENGINEERING
2005

Abstract

In this thesis we propose and explore the Overlay Distribution Network (ODN) concept as an alternative, more cost effective and more flexible deployment approach for Content Distribution Networks (CDN). Currently CDNs are established by deploying a dedicated server infrastructure spanning the Internet, which is prohibitively expensive. With ODN, we propose to lease transport and server resources from server and network providers to establish an overlay network of virtual servers and connecting virtual links, which is then used to deliver content in a similar manner to traditional CDNs. Such deployment strategy is expected to be less costly and could allow content network topologies and capacities to be adjusted on-the-fly according to demand. The aim of this thesis is to address the research issues that arise as a result of this new deployment approach. One of the major issues considered is ODN provisioning, which involves topology planning, resource dimensioning and content replica placement. ODN provisioning is significantly different from and more complex than its traditional CDN counterpart due to the nature of the shared infrastructure environment. ODN provisioning would not only have a different optimization objective but also require topology planning, resource dimensioning and content replica placement problems, which are currently addressed independently, to be addressed jointly.

To address this ODN provisioning challenge, we develop a provisioning framework that could be used to formulate ODN provisioning models that meet these new requirements. ODN provisioning models for a number of key content distribution applications, including web content distribution, pay-per-view content distribution and live streaming multimedia distribution, are then developed and studied. The models are formulated as mixed integer linear programming optimizations that aim to determine the optimal ODN topology, capacity and content replication pattern, which

deliver satisfactory service performance at the minimum cost or maximum profit.

As the above ODN provisioning models belong to the NP (Non-deterministic Polynomial) class of problems, they have extremely high complexity and cannot be solved efficiently for realistically large networks. To tackle this difficulty we develop heuristics that aim to find near optimal solutions with less computation effort. Experimental results show that our proposed heuristics are efficient and able find solutions reasonably close to the optimal (within 36% for the web ODN provisioning problem and 20% for the pay-per-view and live streaming multimedia ODN provisioning problems). Our study also demonstrates that provisioning could become significantly inefficient if the heuristics were not designed properly. For example in the web content ODN provisioning problem, a greedy heuristic adapted from existing CDN replica placement heuristics could produce ODN topologies up to 2.5 times more costly compared to a Lagrangian heuristic designed based the problem formulation structure.

As part of the ODN provisioning study, we also explore the use of content clustering within the ODN provisioning process. By grouping similar content items together into clusters, which are then considered as a single item during provisioning, content clustering would help reduce provisioning complexity and allow the provisioning models to handle problems with significantly larger number of content items. We show that clustering methods previously developed for CDNs do not work well in the ODN environment. Thus we propose a new hierarchical clustering scheme, where content items are clustered based on first delivery resource requirements and then spatial demand distribution. Experimental results demonstrate that this clustering scheme has significant performance improvements over the existing ones.

In this thesis we also look into the future and study the ability of the ODN to support applications that require QoS network paths among servers. To enable better support for such applications, we propose to enhance the ODN architecture with switching capabilities that allow ODN owners to control the flow of traffic within their ODN backbone. We examine and demonstrate the benefits of such capabilities using both quantitative and qualitative studies and then develop a shared switch architecture that could be used to provide such support in a shared infrastructure environment.

Statement of Originality

This is to certify that the work described in this thesis is entirely my own, except where due reference is made in the text.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Thành Vinh Nguyễn

22 August, 2005

Acknowledgments

I would like to thank my supervisors, Professor Farzad Safaei and Dr Paul Boustead, for their dedication, their much appreciated guidance, help and encouragement. I would like to thank Professor Joe Chicharo, who has opened the door for me to pursue my dreams. I would also like to thank Dr Chun Tung Chou, whose guidance during the first stage of the PhD was crucial in shaping the course my research.

I am also very grateful towards the Cooperative Research Center for Smart Internet Technologies, who has funded this work.

I would like to thank my family: my parents, brothers and sisters, whose thoughts are always with me, my other half, Pei Juan Chen, whose love and support has made this all possible.

Last but not least, I would like to express sincere appreciation and gratitude to all members of TITR lab for their support and friendship. It has been a very memorable experience.

*Những cộng trừ nhân chia
Đẫn con đi mãi miết ...*

*plus minus divide and multiply (the study of science)
forever lead you on the journey of knowledge and discovery*

Mom's poem

Contents

1	Introduction	1
1.1	Background	1
1.2	Thesis Objectives	4
1.3	Thesis Outline	7
1.4	Publications	9
2	Literature Review	11
2.1	Introduction	11
2.2	Content Distribution Network Overview	11
2.2.1	Driving Force and Basic Concept	11
2.2.2	CDN Deployment and Operation	14
2.3	Content Distribution Internetworking	16
2.4	CDN Provisioning	18
2.4.1	Provisioning CDN for Web Content	19
2.4.2	Provisioning CDN for Multimedia Content	24
2.5	CDN Content Consistency	26
2.6	CDN Request Routing	28
2.7	Concluding remarks	30
3	An ODN Provisioning Framework	33

3.1	Introduction	33
3.2	ODN Concept	34
3.3	ODN Provisioning versus CDN Provisioning	36
3.4	ODN Provisioning Framework	38
3.5	Conclusion	40
4	Provisioning ODN for Web Content Distribution	41
4.1	Introduction	41
4.2	Provisioning Problem Formulation	43
4.2.1	Input Parameters	43
4.2.2	Decision Variables	44
4.2.3	Optimization Model	45
4.2.4	Problem Complexity	47
4.3	Lagrangian Heuristic	48
4.3.1	Relaxing the Problem	49
4.3.2	Solving Subproblems	50
4.3.3	Constructing a Feasible Solution	52
4.3.4	Subgradient Procedure	52
4.4	Two-Level Greedy Search Heuristic	53
4.4.1	Analysis	53
4.4.2	Heuristic Outline	54
4.5	CDN-based Greedy Search Heuristic	56
4.6	Numerical Results	58
4.6.1	Performance of CDN-based Greedy Heuristic	58
4.6.2	Performance of Lagrangian and Two-Level Greedy Heuristics	60
4.7	Conclusion	63

5	Provisioning ODN for Pay-Per-View Content Distribution	65
5.1	Introduction	65
5.2	Provisioning Problem Formulation	67
5.2.1	Assumptions	67
5.2.2	Input Parameters	67
5.2.3	Decision Variables	69
5.2.4	Formulation	69
5.2.5	Problem Complexity	70
5.3	Alternative Provisioning Models	72
5.4	Heuristic Procedure	76
5.4.1	Applying Lagrangian Relaxation	76
5.4.2	Constructing a Feasible Solution	78
5.4.3	Subgradient Procedure	80
5.5	Heuristic Results	81
5.5.1	Observations	81
5.6	Conclusion	85
6	Provisioning ODN for Live Multimedia Content Distribution	87
6.1	Introduction	87
6.2	Provisioning Problem Formulation	90
6.2.1	Input Parameters	90
6.2.2	Decision Variables	91
6.2.3	Formulation	92
6.2.4	Problem Complexity	95
6.3	Heuristic Procedure	96
6.3.1	Applying Lagrangian Relaxation	96

6.3.2	Solving the Subproblems	99
6.3.3	Constructing a Feasible Solution	101
6.3.4	Subgradient Procedure	103
6.4	Online Optimization	104
6.5	Numerical Results	107
6.5.1	Provisioning Model Behavior	107
6.5.2	Heuristic Performance	108
6.5.3	Online Optimization	112
6.6	Conclusion	115
7	Improving ODN Provisioning Scalability with Content Clustering	118
7.1	Introduction	118
7.2	General Clustering Framework	120
7.3	Content Clustering in ODN Provisioning	123
7.3.1	Analysis	123
7.3.2	Deficiencies in Existing Clustering Metrics	124
7.3.3	Modifying the Clustering Process	129
7.4	Conclusion	132
8	Networking Support in ODN	135
8.1	Introduction	135
8.2	Why Switching Support?	136
8.3	Switching and Bandwidth Efficiency	139
8.3.1	Network, Cost and Bandwidth Demand Models	140
8.3.2	Link Provisioning Models	141
8.3.3	Numerical Results	146

8.3.4	Remarks	150
8.4	Providing Switching Support in a Shared Infrastructure Environment	150
8.4.1	The Challenge	150
8.4.2	The Proposed Solution	151
8.4.3	A Shared Switch Architecture	153
8.5	Conclusion	159
9	Conclusions	161
9.1	Overview	161
9.2	Major Conclusions	163
9.2.1	ODN Provisioning Framework	163
9.2.2	Provisioning ODN for Web Content Distribution	164
9.2.3	Provisioning ODN for Pay-Per-View Content Distribution	165
9.2.4	Provisioning ODN for Live Multimedia Content Distribution	165
9.2.5	Improving ODN Provisioning Scalability with Content Clus- tering	167
9.2.6	Networking Support in ODN	168
9.3	Summary of Major Contributions	169
9.4	Future Work	170
A	Lagrangian Multiplier Update Procedures	182
A.1	Introduction	182
A.2	Adjusting Multipliers in the Web ODN Provisioning Problem	182
A.3	Adjusting Multipliers in the Pay-Per-View ODN Provisioning Problem	183
A.4	Adjusting Multipliers in the Streaming ODN Provisioning Problem	184
A.5	Other Notes	185

B Solving the Subproblems	187
B.1 Solving subproblems Q_1 (5.10) and Q_2 (5.11)	187
B.2 Solving subproblem S_3 (6.17)	189
C Estimating Minimum Number of Replicas	193

List of Figures

2.1	Web content caching	12
2.2	Web server clustering	12
2.3	Delivering content with Content Distribution Networks	14
2.4	A DNS request routing example	15
3.1	An Example ODN Topology	34
3.2	ODN Provisioning Framework	39
4.1	Comparing ODN costs by different provisioning heuristics in the presence of different cost variations	59
5.1	Impact of ODN provisioning model selection on pay-per-view content distribution profit	74
5.2	Comparing different content ranking methods for problems with progressively decreasing server capacities	84
5.3	Solution bounds for the first 1000 iterations	85
6.1	Overlay topology changes in reaction to resource cost variations: (a) high bandwidth and low server cost (b) server cost $\times 10$ (c) server cost $\times 20$	108
6.2	Normalized heuristic solutions obtained with different solution construction methods plotted against link cost multipliers, for 60-node graphs with different link density	110

6.3	Normalized heuristic solutions obtained with different solution construction methods plotted against server cost multipliers, in a graph with link probability 0.9	111
6.4	Normalized heuristic solution obtained with <i>simple paths</i> procedure, with normalized number of established proxies super-imposed	111
6.5	Normalized total service utility by different online optimization methods vs. demand variation created by different types of demand fluctuations	116
7.1	Clustering results for different clustering metrics in the ODN provisioning problem for web content with random demand patterns (lower is better)	125
7.2	Clustering results for different clustering metrics in the ODN provisioning problem for web content with global demand patterns (lower is better)	126
7.3	Results for spatial demand clustering in the ODN provisioning problem for pay-per-view content with different resource requirement variations (higher is better)	127
7.4	Results for spatial demand clustering in the ODN provisioning problem for pay-per-view content with different revenue variations (higher is better)	128
7.5	Results for spatial demand clustering in the ODN provisioning problem for live streaming content with different resource requirement variations (lower is better)	129
7.6	Comparing the performance of clustering by original spatial demand distribution and weighted demand distribution in ODN provisioning for pay-per-view content, where revenue is randomly generated between 1 and 50 (higher is better)	130
7.7	Hierarchical clustering in pay-per-view ODN provisioning problem (higher is better)	132
7.8	Hierarchical clustering in live streaming ODN provisioning problem (lower is better)	133
8.1	Example transport options for ODN backbone connectivity	137
8.2	Bandwidth cost with application flow switching	147

8.3	Bandwidth cost with server flow switching	148
8.4	Switched/non-switched bandwidth cost ratio achieved with different Best-effort:QoS traffic volume ratios	149
8.5	Shared switching resource concept	152
8.6	A Shared Label Switch Architecture	153
8.7	Example switching operations	155
8.8	Implementing a shared switch by extending an existing MPLS switch	158

List of Tables

4.1	Parameters and Variables for Web Content ODN Provisioning Model	45
4.2	Numerical Results for Web Content ODN Provisioning	62
5.1	Parameters and Variables for Pay-Per-View ODN Provisioning Model	71
5.2	Numerical Results for Pay-Per-View Content ODN Provisioning . .	82
6.1	Parameters and Variables for Streaming ODN Provisioning Model .	93
6.2	Numerical Results for Streaming Content ODN Provisioning	113

List of Abbreviations

AS	Autonomous System
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
CDN	Content Distribution Network
CDI	Content Distribution Internetworking
CIG	Content Internetworking Gateway
CPU	Central Processing Unit
DNS	Domain Name Service
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
L2TP	Layer 2 Tunnelling Protocol
LSP	Label Switched Path
MPLS	Multiprotocol Label Switching
NP	Non-deterministic Polynomial
ODN	Overlay Distribution Network
OSPF	Open Shortest Path First
POP	Point of Presence
RAM	Random Access Memory
RFC	Request for Comments
RTT	Round Trip Time
QoS	Quality of Service
TCP	Transmission Control Protocol

TTL	Time To Live
UDP	User Datagram Protocol
VoD	Video on Demand
VPN	Virtual Private Network

Chapter 1

Introduction

1.1 Background

The spectacular growth of the Internet over the last decade has seen it becoming a major content distribution channel, with profound effects on many aspects of our lives, from the way we conduct business to the way we learn, socialize and entertain. Internet based content distribution has therefore become both a big business and a hot field for technological innovation, research and development.

The focus of study in this thesis, Content Distribution Networks (CDN), is among the most recent and innovative in a long line of efforts to improve Internet content distribution scalability. Service scalability has been a major challenge for Internet content services, due to the presence of a large, globally distributed and highly dynamic user population. With inadequate resources, unforeseen demand surges as in the “flash-crowd” phenomenon can overwhelm servers, congest network links and lead to degraded service quality or even total service disruption (Barford and Plonka, 2001). Yet dimensioning the service for peak demand, which could be many times higher than normal level, would either be impractical or uneconomical.

Before the advent of CDN, techniques available to make content distribution scale include content caching and server clustering. Content caching helps reduce bandwidth cost and improve access latency when used at the client side, and helps reduce server computation load and response time when used at the server side. Server clustering, on the other hand, allows multiple servers to share the load and thus helps

scale server resources. These existing techniques, while improving service scalability, have their own limitations and does little to address the need for peak-demand dimensioning.

CDNs were designed to complement existing techniques by replicating content in a network of geographically distributed surrogate servers. Through a request redirection process, requests from end users are redirected to a selected server from which content is delivered. Server selection is a dynamic process and may be based on client-server proximity, network condition and server loads. This allows CDNs to improve access latency, mitigate server overload and avoid network congestion.

The emergence of CDNs has created an innovative business model, where CDN owners deploy and operate the CDN infrastructure, which include large numbers of servers in world-wide locations, and deliver content to end users on behalf of content providers. Delivering content through CDNs often improves access latency, service availability and responsiveness for end users. It also relieves content providers of the need to dimension their servers for very high demand peaks. With its enormous distributed infrastructure, a CDN is able to absorb large demand fluctuations and handle high demand levels, yet still remains resource-efficient due to the fact that the delivery infrastructure is used to deliver content from many different content providers.

However, although the CDN concept has generally been successful and made significant impacts on the Internet content distribution landscape, we believe there are still significant problems in the way current CDNs are deployed. For a CDN to work well, a large number of servers in geographically distributed locations are needed. As existing CDNs have been created with their own dedicated infrastructure, deploying a CDN amounts to building a huge world-wide private network of servers. For example, at the time of writing this thesis, CDN provider Akamai, Inc. has over 14,000 servers in 1,100 networks in over 65 countries (Akamai, 2005). Such infrastructure deployment is prohibitively expensive and difficult to establish. As a result, currently the CDN market consists of only a relatively small number of main players, which were deployed during the boom time of the technology market, such as Akamai (Akamai, 2005) and Speedera (Speedera, 2005).

With future Internet content expected to include richer multimedia, more functionalities and better quality of service, we believe the *geographically distributed* distri-

bution model of CDNs will remain the key to content service scalability. However, considering the deployment cost and the current technology market climate, we also believe that more cost-effective CDN deployment methods are necessary.

The aim of this thesis, therefore, is to look for an alternative approach that would allow CDNs to be deployed at lower costs, so that new CDNs can be deployed and existing CDNs can be extended without having to build a global server infrastructure.

This same aim was also the driver behind the Content Distribution Internetworking (CDI) concept put forward by Internet Engineering Task Force working group CDI (Day et al., 2003). The concept, which is also known as Content Peering, calls for the formation of large CDNs by inter-networking among different CDNs. Through inter-operation, separately-administered CDNs would benefit from both expanded footprints and content portfolio. Since its creation in 2000, the CDI working group have produced a number of requests for comments (RFCs), defining the necessary protocols for CDN inter-operation. However, to the best of our knowledge, the working group have recently concluded without any visible impacts on the CDN industry.

In this thesis, we propose to tackle the problem with an entirely different approach: the deployment of CDNs over *shared virtualized infrastructure*.

There has been an increasing trend towards resource virtualization, sharing and infrastructure outsourcing in the Internet, where services are deployed over leased, virtualized server or transport infrastructure. For example, transport resources are commonly leased from network providers to establish virtual private networks that provide private connections over shared transport infrastructure (Venkateswaran, 2001; Aquino et al., 2000). Enterprise servers commonly support logical partitioning that allows their resources to be virtualized and shared, in order to support entirely different operating systems and services in parallel. Such virtualized servers can often be leased to operate web sites, as in the case of virtual web hosting services, or to perform computation tasks, as with IBM's recent Virtual Linux-on-Demand service (Norton, 2002).

As a continuation of this trend, we believe that shared virtualized infrastructure holds the key to providing future Internet applications with geographically distributed re-

sources cost-effectively. We thus envisage a future Internet environment where network and server resources such as bandwidth, storage space and processing power could be leased from *server providers*, who own physical servers or server farms, and *network providers*, who own physical networks. Instead of deploying dedicated CDN infrastructure, we propose to lease these resources to establish an overlay network of virtual servers and virtual links, which would function as a content distribution network. In this thesis, such a network is called an *Overlay Distribution Networks* (ODN).

The advantages of this new approach would include:

- By using leased resources, ODN deployment would require far less demanding infrastructure investment and resource commitment by the owner of the content distribution application (i.e. the *application provider*, or *ODN owner*) compared to existing CDNs. This would also make it easier for new providers and innovative applications to enter the content distribution market.
- The resource leasing nature of ODNs would also create the possibility of adjusting content network resources on-the-fly. As services might be rolled out without accurate advanced knowledge of the geographical take-up, or only deployed for certain events (e.g. the World Cup Soccer or the Olympics), such flexibility would be highly desirable.

1.2 Thesis Objectives

The ODN approach, however, would mean significant changes to content network deployment and operation, and would introduce new challenges and requirements that have yet to be addressed in current literature. The objective of this thesis is to identify and address these new problems, in order to make the ODN concept a reality. The problems considered include:

1. **ODN Provisioning:** A major part of this thesis is devoted to the issue of *ODN provisioning*. This involves *topology planning*, which determines the desired locations of virtual servers and links, *resource dimensioning*, which determines

the amount of bandwidth, storage space and processing power to be leased at each location, and *replica placement*, which determines where each content item, for example a web page or a video clip, should be replicated. These issues are of paramount importance and need to be carefully optimized because they dictate the performance and resource efficiency of the ODN.

In existing research literature, there is already a significant number of studies on topology planning and replica placement algorithms for traditional CDNs. However, these algorithms are no longer appropriate when considered in an ODN, as the *resource-leasing* nature of this approach would make the problems substantially more challenging and different from what has been considered in the traditional CDN context. This is due to the following reasons:

- In current CDN literature, topology planning and replica placement are usually addressed as separate, stand-alone problems. In the resource leasing environment of ODN, however, topology planning, resource dimensioning and replica placement are interconnected issues and have to be addressed jointly.
- With the ODN approach, leased resources incur costs, thus the goal of ODN provisioning would be to deliver content at the minimum resource cost while conforming to certain service level agreements. In comparison, existing CDN replica placement algorithms usually assume an established infrastructure with certain available resources, which can then be used up in search of the best performance possible (Kangasharju et al., 2001a; Venkataramani et al., 2001).
- Today's Internet content distribution is dominated by web content. However, as future Internet content distribution evolves, we expect to see the emergence of more diversified services with different content types, features, policies and quality of service.

As a result, depending on the service characteristics, different requirements and constraints may need to be considered in the ODN provisioning problem. Currently, although commercial content distribution services have recently emerged (Akamai, 2005; EdgeStream, 2005) that specialize in content types other than web documents, existing CDN place-

ment literature is still mostly built on web content assumptions.

In this thesis, we address these shortcomings by formulating ODN provisioning models that address topology planning, resource dimensioning and replica placement issues jointly, for a number of different applications deemed to be important in the future Internet content distribution industry. With progressively increasing complexity, the applications considered include normal web content, pay-per-view entertainment content and live streaming multimedia. As these problems have very high complexity and cannot be solved efficiently, heuristic procedures are then designed to find approximate solutions within a reasonable time.

2. **Overlay networking support:** In this thesis we also consider the ability of ODNs to support future content applications where inter-server transmission paths better than best-effort are needed, for example in the distribution of interactive content or high definition video content. We propose to enable better support for these applications by enhancing the functionality of the ODN backbone with switching capability at server sites. This would allow application providers to control and manipulate the forwarding of inter-server traffic flows within their overlay topologies, including switching flows between different virtual links and sending traffic along indirect overlay paths instead of relying solely on directly provisioned links or Internet routing.

As will be demonstrated later in Chapter 8, such capabilities would be extremely beneficial in allowing application providers more flexibility in establishing transmission paths with quality of service (QoS) requirements. It would also allow more efficient resource usage in the ODN backbone links through better flow multiplexing.

Apart from demonstrating the advantages of such switching support, we will also look into the challenges of enabling this capability in a shared infrastructure environment and propose a suitable approach using shared switching resources. A shared switch architecture using label switching technologies will also be proposed and possible implementation directions will be discussed.

1.3 Thesis Outline

The organization of this thesis is as follows. In Chapter 2 a critical review of current literature in CDN will be presented. This chapter will start by introducing the key CDN concepts, including the driving force behind its development, its basic structure and operation. Existing works in major CDN research areas will then be described, including Content Distribution Internetworking (CDI), CDN replica placement and topology planning, CDN content consistency and CDN request routing. Among these, CDN replica placement and topology planning is the area most closely related to the work carried out in this thesis and will be examined more closely. Chapter 2 will conclude with a list of shortcomings that we have uncovered in current literature. By analyzing the characteristics of the shared infrastructure environment in the ODN approach and comparing with existing works in CDN literature, Chapter 3 will pinpoint the major differences between provisioning problems in traditional CDNs and ODNs. A suitable provisioning framework will then be developed, which specifies how provisioning and content placement problems should be carried out in the ODN environment. This framework will be used as a guideline for the development of application-specific provisioning models in the next few chapters.

In Chapter 4, the ODN provisioning problem for web content distribution will be studied and formulated as a mixed integer programming problem. As this problem can be proven to be an NP-hard problem and cannot be solved efficiently for realistically large networks, we will then develop heuristic procedures that aim to find near optimal solutions efficiently. Two different heuristics will be developed and evaluated for this problem, including a Lagrangian relaxation heuristic and a two-level greedy search heuristic. We will show that the heuristics are capable of finding solutions close to optimal within reasonable computation time. We will also show that these heuristics, which are designed based upon the problem structure, significantly outperform a greedy heuristic designed based upon existing CDN placement strategies. Various aspects of this work were published in (Nguyen et al., 2003a) and (Nguyen et al., 2003b).

In Chapter 5, the ODN provisioning problem will be considered for another type of content application - the distribution of pay-per-view content. Our work focuses on

the pay-per-view distribution of static content that could be replicated, for example recorded TV programs or movies. As we will point out, although this type of content application has been the subject of a number of research publications and has even seen some commercial adoption, its deployment in a CDN or ODN environment has yet to be studied. In this chapter we will identify the impacts of the pay-per-view distribution model on the ODN provisioning process and formulate a suitable provisioning model, which is also an NP-hard mixed integer linear programming problem. An efficient Lagrangian heuristic for this model will then be developed and evaluated. This work is published in (Nguyen et al., 2005).

Chapter 6 will address the provisioning problem for ODNs that distribute live streaming multimedia content. A provisioning model will be formulated to capture the characteristics and requirements of this application. In this chapter we will also propose to implement application layer multicast in the ODN backbone to distribute live content from content sources to ODN servers, which act as content streaming and processing proxies. In this scenario the provisioning process has to determine not only the ODN topology and content processing/streaming locations, but also how content distribution trees are created and how live content is distributed through the ODN backbone. An efficient Lagrangian heuristic will be developed to find near optimal solutions for this problem.

Apart from tackling the provisioning problem, in Chapter 6 we will also develop a separate online optimization process that is to be used on short time scales in between re-provisioning intervals. This online optimization aims to fine-tune the existing distribution process when demand fluctuates in order to better utilize the currently allocated resources. Three different online optimization schemes will be developed and evaluated, including a full optimization and a partial optimization, both of which are centralized algorithms, and a local optimization, which is a simple distributed algorithm to be performed individually at each server. It will be shown that together with the ODN provisioning model, the online optimization models form a useful tool, which can be used to create optimal topologies which can also handle large variations in service demand. The major part of this work was published in (Nguyen et al., 2004).

In Chapter 7, we will explore the use of content clustering to further improve the

ODN provisioning models developed above. By grouping similar content items into clusters, which are then treated as a single item by the provisioning process, the number of items is reduced, as is the provisioning complexity. Although clustering for CDN content replication has previously been addressed in (Chen et al., 2003), we will show in this chapter that the existing clustering metrics are inadequate for the ODN provisioning models as they fail to take into account other significant factors that could affect clustering efficiency, including content delivery resource requirements and revenue. We will then propose and evaluate a modified hierarchical clustering scheme that is designed to rectify these deficiencies. Part of this work was published in (Nguyen et al., 2005).

In Chapter 8, we will consider the ability of ODNs to support future content applications where inter-server transmission paths with strict QoS requirements are needed. In order to enable better support for such applications, we propose to enhance the functionality of the ODN backbone with switching support that allows application providers to control the forwarding of traffic within their overlay networks. In this chapter we will first demonstrate the benefits of such switching support through both qualitative and quantitative studies. We will then consider the available options for providing such support in ODN and propose a solution based on shared switching resources. A suitable shared switch architecture will also be identified and described. Finally, Chapter 9 will present a summary of major conclusions from these studies, a summary of contributions for the whole thesis as well as possible directions for future work.

1.4 Publications

Nguyen, T. V., Safaei, F., and Boustead, P., “An Architecture for Carrier Grade Programmable Networks”, *In Proc. Global Telecommunications Conference 2002. GLOBECOM '02*, Taipei, Taiwan, November 2002, pp.2016-2020.

Nguyen, T. V., Chou, C., and Boustead, P., “Provisioning CDN over Shared Infrastructure”, *In Proc. IEEE ICON2003*, Sydney Australia, September 2003, pp.119-

124.

Nguyen, T. V., Chou, C., and Boustead, P., “Resource Optimization for CDN over Shared Infrastructure”, *In Proc. Australian Telecommunications Networks and Applications Conference (ATNAC)*, Melbourne Australia, December 2003.

Nguyen, T. V., Safaei, F., and Boustead, P., “Provisioning Overlay Distribution Networks for Live Streaming Media”, *In Proc. Australian Telecommunications Networks and Applications Conference (ATNAC)*, Sydney Australia, December 2004.

Nguyen, T. V., Safaei, F., Boustead, P., and Chou, C. T., “Provisioning Overlay Distribution Networks”, *Computer Networks*, 49(1), September 2005, pp.103118.

Chapter 2

Literature Review

2.1 Introduction

This chapter introduces the current literature in Content Distribution Networks (CDN). As a background for better understanding of both previous studies and those carried out in this thesis, the chapter will start with an overview of existing CDN, their driving force, architecture and operation. Key CDN research issues and related works in existing literature will then be reviewed, including works in Content Distribution Internetworking (Section 2.3), content and server placement (Section 2.4), request routing (Section 2.6), and content consistency management (Section 2.5). This chapter will finally conclude with a discussion of shortcomings in current literature and how some of the issues have been addressed in this thesis.

2.2 Content Distribution Network Overview

2.2.1 Driving Force and Basic Concept

The major driver behind the development of CDN was the need to scale web content delivery to cope with a large, globally distributed and highly dynamic user population. The more popular a web site is, the more challenging this problem becomes. Before the emergence of Content Distribution Networks, the main techniques for scaling web content delivery could be grouped into caching and server clustering.

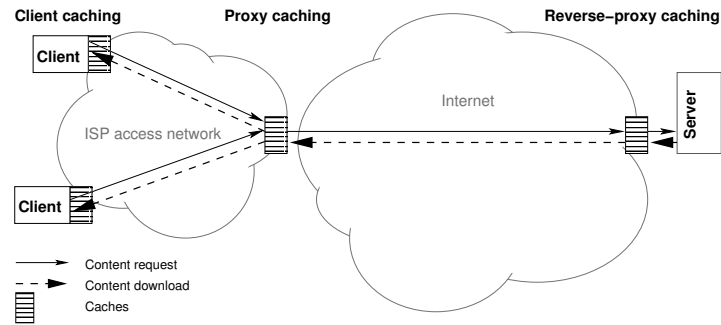


Figure 2.1 Web content caching

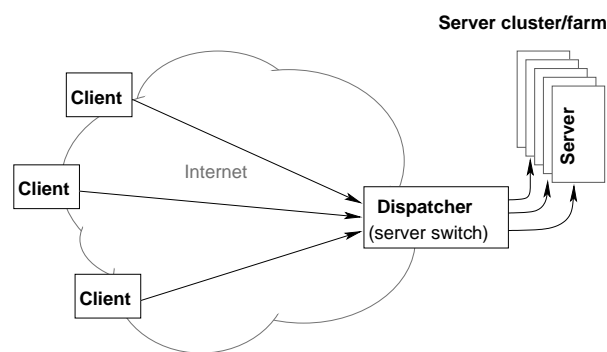


Figure 2.2 Web server clustering

With caching, content caches are deployed to store selected web pages that have been requested by users. These cached copies can be readily reused to satisfy new repeated requests, thus improving response time and reducing bandwidth usage. There are many different variations with different cache location, arrangement and operation, including *client caching* performed by individual users' web browsers, *proxy caching* deployed by ISPs to serve all users in a network or *reverse proxy caching* used at server side to cache frequently requested web pages and reduce server processing load (Barish and Obraczke, 2000; Dilleya and Arlitt, 1999). Caches could also be deployed as independent stand-alone elements or organized hierarchically into cooperative tree-like structures with *hierarchical caching* (Chankhunthod et al., 1996). These different techniques are illustrated in Figure 2.1.

The server clustering technique, on the other hand, scales server resources by de-

ploying multiple servers with the same content in a *server cluster*, which is sometimes called a *server farm*. The cluster is typically connected to the Internet through a front-end *dispatcher* or *server switch* that intercepts and examines incoming requests and allocates each to one of the available servers (Chase, 2001). From users' perspective, the cluster thus appears to be a single, powerful machine (Figure 2.2). This technique allows massive server resource deployment, load balancing among servers, and ability to remove or add servers without bringing the service offline.

The above techniques, however, have the following limitations:

- Caching may perform poorly if requests for a particular object, while large in aggregate, are spread thinly across caches and uncooperative caching may lead to unnecessary storing of the same set of objects in many different locations.
- Server clustering, on the other hand, improves server scalability but does little to avoid problems originated from inside the network, such as congestion or high client-server latencies.
- Additionally, although server clustering allows more server resources to be deployed, determining the right amount of resources is a difficult task. With inadequately dimensioned servers, unforeseen demand surges as in the “flash crowd” phenomenon (Barford and Plonka, 2001; Jung et al., 2002), could overwhelm servers, congest the network link and lead to degraded service quality or even total service disruption. Yet dimensioning for the highest possible demand, which could be many times higher than normal and happens only from time to time, may not always be practical or economical.

Against this background, CDNs emerged as a solution that both further improves web service scalability and relieves content providers of the server dimensioning burden. A CDN typically has a large number of servers in distributed locations around the world, which are deployed and operated by a CDN provider. The servers are used to deliver web content to end users on behalf of content providers, who would buy the delivery service from the CDN provider. Content items, such as web pages, images, audio and video files, are replicated to multiple locations close to their user base and

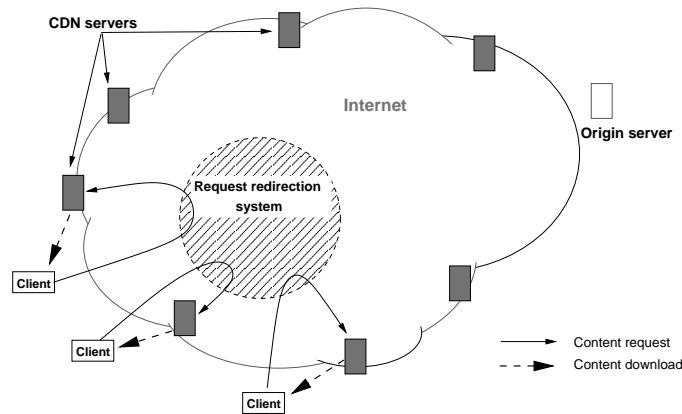


Figure 2.3 Delivering content with Content Distribution Networks

each content request is intercepted and routed to a server considered the most suitable instead of the origin server (see Figure 2.3). By pushing content closer to users and performing request routing with the knowledge of server-user proximity, content availability, server load and network conditions, a CDN would be able to mitigate server overloading, bypass network congestion and improve access latency and service availability. Delivering content through a CDN would also allow a web site to cope with large demand fluctuations effectively without content providers having to dimension their servers for very high demand peaks. With CDNs, Internet content delivery has moved from the traditional *client-server* model to a *client-distributed network of servers* model.

At the time of writing this thesis, prominent commercial CDN providers include Akamai (Akamai, 2005) and Speedera (Speedera, 2005), Accelia (Accelia, 2005) and EdgeStream (EdgeStream, 2005), and example content providers that use CDN services include Yahoo, Reuters, Red Hat, Apple Computers, CCBN, Victoria's Secrets and IBM Corporation.

2.2.2 CDN Deployment and Operation

Current CDN service providers established their CDNs by deploying large numbers of dedicated CDN servers around the world. These servers are typically placed in the networks of major Internet Service Providers (ISP), who agree to house the servers

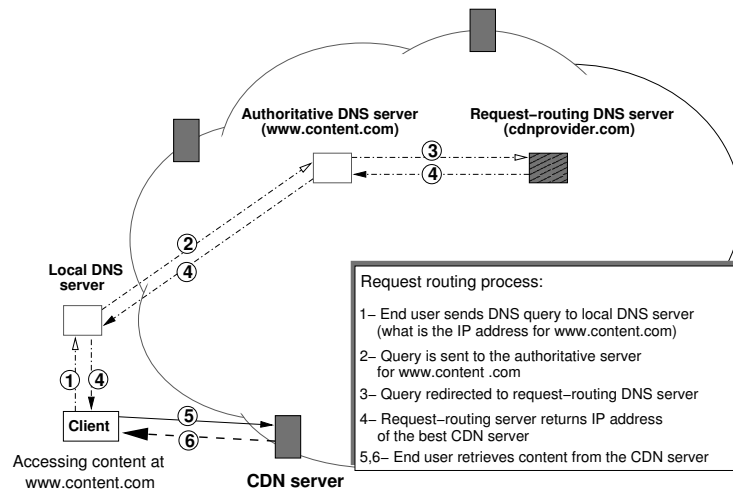


Figure 2.4 A DNS request routing example

in exchange for potential Internet bandwidth usage reduction due to their presence. The servers, however, are still owned, operated and maintained by the CDN owner. In order to intercept and redirect content requests from end users to suitable CDN servers, a *request routing* system, which is also called a *request redirection* system, is used. In current CDNs, this system is most commonly based on a modified Domain Name Service (DNS). DNS is used in the Internet to translate human-readable web addresses (such as *www.yahoo.com*) into IP addresses that computers need to connect to web servers (such as *130.130.10.100*). This address resolution is done transparently, behind the scene, by the web browser and the operating system every time a user accesses a website. In DNS-based request routing, a specialized request-routing DNS server operated by the CDN provider is inserted into the resolution process. This server is capable of returning different answers to each DNS query based on policies that may take into account client location, available servers, server proximity, server load and network condition. As a result, different users accessing the same web address may be given different IP addresses and thus connect to different CDN servers.

A DNS request routing example is shown in Figure 2.4, illustrating the steps that take place when an end user accesses a website at *www.content.com*, which is delivered through a CDN. Note that this example is used to illustrate the main idea and

thus shows just one of the many different DNS request routing variants that could be used in practice. For example, it is also possible to have multiple request-routing servers involved in a single DNS resolution in order to implement more elaborate request routing policies. In that case a higher level DNS server may receive and then direct DNS queries to lower level request-routing DNS servers that are located in the enquiring hosts' regions. Each lower level server is in charge of a region (e.g. country or continent) and thus would resolve the query with more accurate information on nearby servers and network conditions. This can be done by having the higher level DNS server return either a CNAME record (redirect the query to a new domain name) or a NS record (redirect the query to a new name server) instead of IP addresses, in response to a DNS lookup (Barbir et al., 2003).

Although it is technically possible to reroute content requests using other methods, such as network layer anycast (Agarwal et al., 2001), transport layer or application layer redirection (Barbir et al., 2003), DNS-based request routing is still most commonly used in existing commercial CDNs due to the ubiquitous availability of DNS services and the ability to route requests transparently, without modifying existing web server or client architectures.

Depending on the service level agreement between the content provider and CDN owner, a web site may have all or only part of its content replicated to CDN servers. In the latter, heavily requested graphic or video files are typically replicated while container web documents are still downloaded from the origin server, with references to replicated content items modified to point to a domain under the control of the CDN provider. Subsequent requests for these objects would then go through the normal request routing system as described above. However, as shown in a study by (Kangasharju et al., 2001b) on performance of both simulated and real life CDNs, such arrangement typically results in much longer latencies, due to extra DNS resolutions, compared to full site replication.

2.3 Content Distribution Internetworking

The effectiveness of a CDN depends on having servers close to the user base of their content. Thus for a CDN to work well, either highly accurate information on demand

distribution or a large server infrastructure covering the Internet is needed. As a result, current CDN deployment typically requires in the order of thousands of servers in world-wide locations and is prohibitively expensive.

In search of a more cost effective CDN deployment strategy, the Content Distribution Internetworking (CDI) concept was proposed, pioneered by Internet Engineering Task Force (IETF) CDI Working Group, which was set up to investigate the technology requirements, feasibility and other aspects of the CDI direction. With CDI, it is envisaged that large Internet-wide CDNs could be formed by internetworking among smaller CDNs that belong to different operators, which allows individual CDNs to share resources, increase capacities and extend their footprint as well as content portfolio.

The CDI working group proposed to achieve this internetworking by creating standardized content internetworking gateways (CIG), through which peering relationships could be established among the corresponding components of partner CDNs. These components include the request-routing infrastructure that redirects users to suitable CDN servers, the distribution infrastructure that manages content replication and the accounting infrastructure that tracks and collects data on request-routing, content distribution, and delivery within each CDN (Day et al., 2003). As a result, Content Distribution Internetworking is also termed Content Peering or CDN Peering.

A number of informational Request For Comments (RFC) have been published by the group, covering CDI architecture overview (Green et al., 2002), possible models and usage scenarios (Rzewski et al., 2003), terminologies and requirements (Day et al., 2003) and a review of known request routing techniques that could be used in a CDI environment (Barbir et al., 2003). However, no further development has been done and to the best of our knowledge the CDI Working Group has recently concluded without visible impacts on the existing CDN industry.

Apart from the efforts of the CDI Working Group, a small number of other works in the CDI direction have also been published in research literature, including those by (Biliris et al., 2001), (Biliris et al., 2002), (Vuong et al., 2002), and (Turrini, 2004). In (Biliris et al., 2001) and (Biliris et al., 2002), the authors proposed a CDN brokerage architecture where one *brokering* CDN establishes peering relationships

with and collects information on other partner CDNs. The DNS-based request redirection system of this brokering CDN will be able to decide whether to serve a request internally using its own servers or redirect to one of the partner CDN, depending on each CDN's coverage and available resources.

In (Vuong et al., 2002), the authors proposed a hierarchical request routing architecture for the CDI environment. The architecture is *hierarchical* in the sense that it is modelled after the IP routing, with separate request routing processes for routes within a CDN, between CDNs and between Autonomous Systems. The authors proposed to achieve request routing efficiency by delegating content routing to the network layer, including making network components aware of content replication and expanding the existing BGP routing protocol to accommodate both IP and name-based (e.g. destination address is *www.content.com* instead of *130.130.10.0/24*) routes. Although this is an interesting approach, it has a number of potential weaknesses. Firstly, it would require significant changes throughout the Internet routing infrastructure. Secondly, with an ever increasing number of content types, content providers and content items entering the Internet content market, there would be an explosion in the number of name-based routes the proposed request routing system has to handle.

Work in (Turrini, 2004), on the other hand, is based on the existing work by the IETF CDI Working Group and aims to contribute towards the definition of a concrete CDI architecture by proposing and evaluating specific protocols for information exchange in CDI. This includes information exchange between request routing systems, content distribution systems and CDN accounting systems. This, however, has been a work in progress and its completion status is unclear at the time of writing this thesis.

2.4 CDN Provisioning

For each CDN, how provisioning issues, including topology planning, replica placement and resource dimensioning, are addressed has great bearing on the efficiency and quality of service it delivers. Yet with typical CDNs having a large number of servers to manage and a large number of content items to replicate, this is undoubtedly a challenging task. Although the information on how exactly this is done

in current commercial CDNs is not publicly available, there has been a significant number of publications in research literature that tackle this problem.

Existing CDN provisioning literature could be roughly divided into two categories - those considering the problem in a traditional web-content environment and those looking at future CDNs that focus on multimedia content.

2.4.1 Provisioning CDN for Web Content

A significant portion of existing CDN provisioning studies were devoted to the placement of web content replicas. Among the earliest and most influential of these studies were perhaps those published in (Kangasharju et al., 2001a) and (Kangasharju et al., 2002). In these publications the authors considered the problem of replicating a set of content items with known spatial distribution of demand over a set of CDN servers with known location and storage capacities. The aim of this placement was to minimize the average distance between users and their requested content. The problem was formulated as a linear optimization model.

To formulate this problem, the authors considered an Autonomous System-level network topology of I nodes, where each node $i \in \{1, 2, \dots, I\}$ represents an Autonomous System (AS), has S_i bytes of storage capacity and a total content request rate of λ_i from its clients. A content population of J items was assumed, with item $j \in \{1, 2, \dots, J\}$ having size b_j bytes and popularity of p_j . Each possible content placement pattern is then described by an array $\mathbf{x} = \{x_{ij}\}$, where $x_{ij} \in \{0, 1\}$ are binary variables indicating if content item j is replicated at AS i or not. Using these parameters, the problem was formulated as follows:

$$\text{Minimize } C(\mathbf{x}) = \sum_{i=1}^I \sum_{j=1}^J s_{ij} d_{ij}(\mathbf{x}) \quad (2.1)$$

subject to:

$$\sum_{j=1}^J b_j x_{ij} \leq S_i, \forall i \quad (2.2)$$

where $s_{ij} = \frac{\lambda_i p_j}{\sum_i \lambda_i}$, $d_{ij}(\mathbf{x})$ is the distance from AS i to the nearest replica of j , under placement pattern \mathbf{x} , and $C(\mathbf{x})$ is the average access distance across all users and all content items, given by this placement pattern.

As this problem is a NP-complete problem, it can not be solved efficiently at large size. As a result the authors developed and evaluated a number of different heuristics to find near optimal placement patterns. It was found that a greedy heuristic, which basically places one replica at a time, and each time chooses the pair of $\langle \text{content item, replicating location} \rangle$ that gives the most reduction in the average access distance $C(x)$, until no further replica could be placed, performed very close to the optimal solution.

A slightly different version of replica placement problem is found in (Venkataramani et al., 2001). In this study, the aim is also to distribute content items to different locations over a Wide Area Network (WAN) so that, given a known demand distribution pattern, the resulting average access distance is minimized. However the placement is subject to the constraint that the replication has to be completed within a certain time period t_{filled} , with each server having a certain bandwidth limit for content uploading, instead of storage space constraint. Nevertheless, the resulting optimization problem is mathematically similar to the space-constrained placement problem in (Kangasharju et al., 2001a). The major difference in this work lies in the fact that the authors assumed an underlying network with hierarchical distance structure, with each node having a set of nearby neighbors all at approximately the same distance, and then a set of next closest neighbors all at approximately the same (but greater) distance, and so on. This structure was exploited to design greedy heuristics in which placements are considered iteratively from one level of the distance hierarchy to another (Venkataramani et al., 2001).

Another different replica placement problem was considered in a study by (Qiu et al., 2001), which was published around the same time as (Kangasharju et al., 2001a). Although having the same goal of minimizing the distance-based access cost, this study assumed a simplified placement strategy where servers containing all content items are replicated and each user simply retrieves all required content from the nearest server. The optimization problem thus aims to place a pre-determined number K of servers so that total access cost is minimized, instead of replicating individual objects as in the above studies. Although this might not be a very practical assumption, the authors expected this to give an insight into the future when falling storage cost makes it possible to replicate everything at each location. This study, therefore, could

also be considered as dealing with the CDN topology planning problem rather than content replica placement problem.

The authors modelled this optimization using the classic K -mean problem (Hochbaum and Shmoys, 1985). Given a network of N potential servers and client locations, with each client j having total request rate λ_j and access cost c_{ij} for retrieving content from server i , the optimization has the following format:

$$\text{Minimize } C(\mathbf{x}) = \sum_{i=1}^N \sum_{j=1}^N \lambda_j c_{ij} x_{ij} \quad (2.3)$$

subject to:

$$\sum_{i=1}^N x_{ij} = 1 \quad (2.4)$$

$$\sum_{i=1}^N y_i \leq K \quad (2.5)$$

$$x_{ij} \leq y_i, \forall i, j \quad (2.6)$$

$$x_{ij}, y_i \in \{0, 1\} \quad (2.7)$$

where binary variable y_i is used to indicate if a server is placed at location i and binary variable x_{ij} is used to indicate if the client at location j is directed to server i .

As this K -mean problem is also known to be NP-complete (Hochbaum and Shmoys, 1985; Charikar et al., 2002), the authors developed and evaluated a number of heuristic solution procedures, including a greedy algorithm that places one server at a time, each time selecting the location that gives the most performance improvement, and a *hot spot* algorithm that places servers at locations with highest total demand in their vicinity. Similar to the previous works, it was found that the greedy algorithm had the best performance, with total access cost very close (within a factor of 1.1-1.5) to the optimal solution.

Apart from work published in (Qiu et al., 2001), there are also a number of other studies that address similar CDN server placement (i.e. topology planning) problems, including (Radoslavov et al., 2001) and (Jamin et al., 2001). In (Radoslavov et al., 2001), the same problem and formulation as in (Qiu et al., 2001) is presented. However, the authors aim to develop a practical heuristic that does not require knowledge of client demand but still performs reasonably well. It is proposed in this study

that servers should be placed at highly connected nodes - i.e. nodes that have high *fan-out* levels. Thus given a number of K servers to place, their locations would be selected from the potential nodes in decreasing order of their fanout levels. The authors hoped that as nodes with larger fan-out levels would probably be the closer on average to other nodes, they also represent good choices for replica locations. It was found in this study that for certain topologies, including random and power law graphs, the algorithm performs very close to the greedy algorithm used in (Qiu et al., 2001). However, the same is not true when network nodes are sparsely connected. In (Jamin et al., 2001), the server placement problem is again considered and modelled using the classic K -mean problem. However, in this study the authors even further simplify the optimization problem by not taking into account any client demand distribution, or server capacity, and simply aim to minimize the average distance between users and their nearest CDN server. As a result, a number of existing heuristics available in Operations Research literature for approximating the solution of this simple problem were used. Instead of evaluating heuristics as in other studies, the authors aimed to investigate the relationship between performance improvements and the increments in the number of servers. The results in this study indicated that the performance gain for each added server diminishes with increasing number of servers and that adding servers is effective in reducing latency and server load only when the number of servers are within a certain threshold. This result is also confirmed in another separate study using simulations with real web server traces, as published in (Li and Liu, 2003).

Apart from the studies that aim specifically to address CDN replica placement or topology planning issues, there are also a number of earlier publications that originated from other research areas, but considered similar problems and thus are often quoted and referenced in CDN literature. Examples include (Jamin et al., 2000), (Li et al., 1999), (Krishnan et al., 2000) and (Korupolu et al., 2001) and (Cidon et al., 2001).

In (Jamin et al., 2000), the problem of placing Internet instrumentation - tracers used to provide a distance map of the Internet from which relative distances between different hosts could be gauged - is considered. As the accuracy of distance estimates depends on having tracers close to the users, the tracer placement problem aims to

minimize the total distances between user locations and their nearest tracer. Given the number of tracers to be deployed, the placement problem becomes another instance of the K -mean optimization problem, similar to the server placement problem formulated in (Jamin et al., 2001).

In (Li et al., 1999) and (Krishnan et al., 2000), on the other hand, the placement of a given number of web proxy caches in order to minimize the total access latencies is considered. Although the problems described are similar and applicable to CDN server placement, the solution procedures are not, as the authors assumed underlying networks with tree topologies, which is then exploited using dynamic programming techniques in solving the placement problems. Similar situations are also found in the remaining studies, which considered the placement of content items in web caches (Korupolu et al., 2001) and in cable television distribution systems (Cidon et al., 2001). In these studies the hierarchical nature of the underlying infrastructure (i.e. caches or cable head-ends) is also exploited to model the network as a tree structure, which again allows the placement problems to be solved efficiently with the use of dynamic programming algorithms.

A survey on content replica and server placement in CDN and related fields, with an extensive list of available works in research literature is presented in (Karlsson et al., 2002). In this study the authors also propose a framework that allows comparison of different placement models defined in existing publications. According to this framework, the placement models are compared in different aspects including the mathematical structure of the problem definition, the types of heuristics developed, the estimated time it takes to solve and solution quality.

In (Karlsson and Mahalingam, 2002), apart from providing a similar comprehensive review of existing CDN replica and server placement literature, the authors also aim to evaluate the feasibility of using regular demand-based caching algorithms instead of proactive replica placement in a CDN. Caching algorithms considered include least recently used (LRU) and popularity caching. It is found that normal LRU caching with evaluation interval in the order of hours/days can achieve similar performance to previous replica placement algorithms. However, as the authors also acknowledged, this result is based on simple models where only server storage capacity is constrained but not server load or bandwidth. Whether these results would

be valid in general is not yet explored.

In a more recent study published in (Chen et al., 2003), another comparison between caching and proactive replication is carried out, which found that replication performs significantly better in terms of number of replicas required to achieve the same access latency. In this same study, the authors proposed and implemented a content clustering framework that allows content items to be grouped and replicated together in order to reduce the number of items and improve the scalability of content placement algorithms. As grouping multiple content items together may result in unnecessary replications, content clustering essentially sacrifices placement solution quality in order to reduce computation time. In (Chen et al., 2003), the authors found that by grouping content items (i.e. web pages) based on either similar popularity or similar spatial access distribution, the number of content items could be drastically reduced without significant reduction in final solution quality. Although this is a very interesting research direction, we have not been aware of any other studies apart from (Chen et al., 2003) that consider content clustering in CDN.

2.4.2 Provisioning CDN for Multimedia Content

With the growth of the Internet, the increasing penetration level of broadband home connections and advances in digital multimedia technologies, the role of multimedia content in Internet content distribution is expected to become more and more important. The desire for Internet-based multimedia content could be seen from the plethora of studies that are dedicated to the topic of Internet Video on Demand (VoD) in research literature, for example those published in (Cleary, 1995), (Acharya and Smith, 2000), (Chan and Tobagi, 2001) or (Tran et al., 2003) and references therein. With the delivery of multimedia content, heavier burdens would undoubtedly be placed on server and network infrastructure than with traditional web content. As a result, service scalability issues would become even more pressing and the use of CDNs as a delivery platform will naturally be an attractive solution. In fact, currently distribution services dedicated to multimedia streaming content have already been offered by some commercial CDN providers (Akamai, 2005; EdgeStream, 2005).

The delivery of multimedia content would create new factors that need to be considered by the replication process, for example video and audio files are typically much

larger than HTML pages or images in traditional web content. Delivery resource requirements for different items could also vary significantly in both bandwidth and computation, as content items could be encoded with different bit rates or require extra processing such as adaptation, personalization, transcoding or compression. As a result, delivery resource consumption has to be taken into account alongside demand distribution, which renders content replication models based on web content seen in previous section unsuitable.

In existing Video on Demand literature there are a number of studies that did consider the problem of content caching/replication in proxy video servers, for example (G. and Tobagi, 2001) or (Wang et al., 2002). However, these works actually do not extend to the CDN environment as the authors typically considered scenarios with a central (origin) server and a proxy server. Other factors vital to CDNs such as topology information, spatial distributions of users, demand and servers are not taken into account.

The amount of existing studies in research literature that specifically address the provisioning of CDNs for multimedia content distribution, on the other hand, has been limited. The studies that we are aware of include those by (Almeida et al., 2002), (Yang and Fei, 2003), (Cahill and Sreenan, 2003) and (Cahill and Sreenan, 2005). In (Almeida et al., 2002), the authors consider the dimensioning of CDN servers and the placement of video content when bandwidth skimming, a VoD scalable transmission scheduling technique using a combination of unicast and multicast channels (Eagera et al., 2000), is used to distribute video content through CDNs. The placement model in this work aims to minimize the total delivery cost by replicating video content to CDN servers using both full file and partial file replication options. The authors formulated a cost model based on bandwidth skimming protocols and performed a thorough study of the total cost over a large CDN configuration space, including different bandwidth cost, demand levels and number of servers. However, in this work only bandwidth cost is considered in the cost model and demand is characterized by a request per server ratio, which effectively assumes a uniform server and demand distribution and does not consider topological information.

The study presented in (Yang and Fei, 2003), on the other hand, aims to examine the effects of the number of CDN servers on the resulting latency of video content

distribution. This was studied using both analysis and simulation with video content cached in CDN servers as full files, using a few variations of the common cache content replacement algorithm Least Recently Used (LRU). The study indicated that there is an optimal number of servers beyond which access latency worsens instead of improving. Although the result seems counter-intuitive, this may be due to the fact that in this study clients are always assumed to connect to the nearest server, which has limited storage and caches content based on LRU algorithm. As a result, more servers means fewer clients per server, which may reduce the effectiveness of the LRU algorithm.

In (Cahill and Sreenan, 2003) and (Cahill and Sreenan, 2005), the authors consider the delivery of high quality video content, such as recorded TV programs and movies using the CDN architecture. A placement process is proposed that allows content replication to be adjusted dynamically on-the-fly in order to minimize delivery cost, including streaming bandwidth, content transfer between servers, storage costs at servers and the cost of running these servers. In order to achieve this, the system continuously monitors and groups users who are from the same location and accessing the same content into *clusters*. Periodically, a dynamic placement algorithm is executed, which considers each $\langle server, cluster \rangle$ pair to determine if any cluster could be better served by another server. This may result in the transfer of content and redirection of clients in the cluster to a more suitable server. The authors also propose to create a flexible pool of servers by leasing idle servers from ISPs for use as streaming servers, which could be turned on and off as required.

2.5 CDN Content Consistency

As content gets replicated away from their origin, it is desirable to keep replicas and origin content items consistent, so that old (stale) replicas are not used to serve client requests if the origin item has already been updated. This task is usually called content consistency management.

According to (Ninan et al., 2002), this is a well studied issue in the context of web content replication with a single cache or proxy, where content consistency is usually managed with techniques such as: (1) caching content with time-to-live (TTL) indi-

cators (Alex, 1992), (2) periodic polling and invalidation between caches and origin server (Cao and Liu, 1997), (3) server-driven content refresh that combines pushing and pulling (Srinivasan et al., 1998) and (4) leases, where for each request from a proxy, the server gives a period of time within which it agrees to notify the proxy if the original content is modified (i.e. a *lease*) (Yin et al., 2001).

A number of studies have then extended these or proposed new consistency maintenance techniques for the multi-server context of CDNs. For example, in (Fei, 2001) the authors propose a hybrid approach that combines server-based refresh (i.e. flooding out new content as it is updated) and invalidation (i.e. sending only an invalidation message). The origin content server decides on which technique to use for each object, based on its demand level in order to minimize update traffic. The intuition is that for highly popular objects, pro-active server-based refresh is beneficial while for less popular objects that would consume unnecessary bandwidth.

In another study by (Bradley and Bestavros, 2002) a caching extension to HTTP called Basis Token Consistency is proposed, which allows a clock value similar to a version number to be associated with each data source and tracked by CDN servers or caches. The authors showed that this approach gives better performance than a traditional time-to-live approach. Work in (Ninan et al., 2002), on the other hand, proposes a scheme called Cooperative Leases that uses a single lease to represent multiple CDN servers and application-layer multicast to propagate server notifications. This work showed that for consistency maintenance in a CDN, significant reduction in server message and state overhead can be achieved in comparison with traditional schemes.

Consistency issues could also affect the efficiency of content replication in CDNs, as placing more content replicas may give better access latency but also leads to higher bandwidth consumption due to content updates and consistency management. However, such a connection has not been considered in literature. This is due to the common assumption that the bandwidth consumption by content replication and updates would be much smaller than by content delivery from CDN servers to clients, and thus is not a significant factor.

In general, consistency management techniques developed for CDN could also be applied to ODN. Hence we do not see the need to study this issue separately for

ODN in this thesis.

2.6 CDN Request Routing

As described earlier in Section 2.2.2, the effectiveness and novelty of the CDN architecture lies in the capability to redirect different users requesting the same content items to different locations based on service provider-defined policies such as server proximity, server load, network condition or content availability. Request routing therefore has become a major CDN research area, with a significant amount of literature looking at both request routing methods and metrics used in making request routing decisions.

Although DNS-based request routing is commonly used in today's CDNs due to its simplicity and transparency and the ubiquity of DNS service, the technique does have its limitations. For example, as pointed out in (Barbir et al., 2003), DNS request routing only allows resolution at the domain level while an ideal request routing system should be able to route requests per item level. DNS resolution results are also usually cached at local DNS servers for a period of time for later use, thus in order to allow quick adaptation to changing network and servers current request-routing DNS servers often mark their DNS query responses with a short time-to-live (TTL) values. This, however, has the side effect of increasing the volume of requests to DNS servers. Furthermore, as DNS queries are usually relayed on behalf of end users by their ISP's DNS servers, a request-routing server system only knows the IP address of a client's local DNS server but not that client's exact IP address in evaluating client-server proximity. As a result, clients have to be assumed to be close to the enquiring DNS server, which might not always be true.

The effects of these drawbacks have been confirmed in a number of studies on real life CDN performance. For example, in (Johnson et al., 2000) the authors measure the performance of a number of current commercial CDNs and show that DNS request routing in these CDNs does not always reliably redirect a client to the nearest replica available. In (Shaikh et al., 2001), the authors show that reducing time-to-live for DNS records may increase the total load on DNS servers by orders of magnitude and that the assumption of close proximity between clients and their DNS servers

may not always hold, which sometimes lead to significantly ineffective request routing decisions.

As a result, there have been a number of attempts in CDN literature that look for alternative request routing approaches. One example is the modified DNS request-routing scheme proposed in (Chen et al., 2004), where the redirection decision is pushed closer to clients using modified local DNS servers. In this scheme, the CDN's request-routing DNS server simply returns a list of all possible servers together with their current load information, and the local DNS server will then acquire extra information such as network latency, route bandwidth, hop count towards each candidate servers using the local Internet gateway router. The information will then be used to make a local decision on the best server to serve a particular request. Although the simulation results in this study have shown better performance in access latency and network bandwidth balancing compared to simpler schemes using round robin server redirection, whether or not it is better than systems already in use in commercial CDNs is not clear. Furthermore, this scheme would require substantial modification to the client side DNS servers, which are usually outside the influence of CDN providers.

Apart from DNS-based solutions, request routing might also be done via network, transport, or application layer support, as pointed out in (Barbir et al., 2003). A number of other works have considered these alternatives, for example a network-layer request routing scheme is proposed in (Agarwal et al., 2001), which uses network layer anycast for request routing. IP *anycast* (Katabi and Wroclawski, 2000) essentially involves assigning different hosts the same anycast IP address and packets sent to this address would be forwarded transparently to the *nearest* anycast host, where *nearest* is in terms of routing metrics defined by the routing process. In (Agarwal et al., 2001), it is proposed that request routing agents (termed "Smart Boxes" by the authors) are deployed at many geographically diversified locations and are assigned the same anycast address space. Thus content requests destined for an anycast address would be transparently routed to the local routing agent, which will forward it on to a suitable server, based on its information on server load and proximity.

Network layer anycast is also used in the request routing scheme proposed in (Miura and Yamamoto, 2002). However, in this work the authors used an active networking

approach where network routers are augmented with application layer functionalities that allows them to monitor content request and retrieval traffic in order to measure application layer round trip time (RTT). The authors argue that by using application layer RTT rather than network RTT, both network and server related latencies are taken into account, which would lead to better user-perceived performance.

The study by (Chen et al., 2002), on the other hand, proposes to establish a peer-to-peer network, based on the Tapestry peer-to-peer architecture (Zhao et al., 2001), among CDN servers. Users simply connect to the nearest server and then content requests are passed along from one server to another until a suitable replica is found. There are also a number of request routing studies that concentrate on the gathering of information used in the routing decision process instead of how to perform the redirection, such as (Andrews et al., 2002) and (Mukhtar and Rosberg, 2003). In (Andrews et al., 2002) a system called Webmapper that is designed to be co-located with CDN servers is proposed. This system passively monitors client connection statistics, such as client IP addresses and connection round trip times, which are then used to aggregate client address spaces and make request routing decisions, in conjunction with a DNS request routing system. In contrast, in (Mukhtar and Rosberg, 2003) active client side involvement in the measurement is proposed. In this approach, when a client connects to a server, the request routing system in that server pushes a probing script (written in JavaScript) into the client's web browser, which silently instructs the client to contact and measure loading times of a number of servers and report back the results. The information is then used to build the system's request routing database.

Again, request routing techniques for CDN would be applicable to ODN and need not be further considered.

2.7 Concluding remarks

This chapter has provided a review of existing CDN technologies as well as current literature in major CDN research areas, including Content Distribution Internetworking, CDN content replica and server placement, CDN content consistency management and CDN request routing. From this review, we have been able to pinpoint

the following shortcomings, which have helped drive the research carried out in this thesis:

- Research into alternative, more cost-effective CDN deployment strategies has been lacking. Alternative deployment strategies are desirable because currently the cost of building a CDN, which essentially involves establishing a world-wide private server infrastructure, is prohibitively expensive. So far the only initiative in this direction has been the Content Distribution Internetworking (CDI) concept. However, this relies on standardization and interoperation among CDN providers, which could be a serious obstacle to its adoption. In fact, the IETF Working Group that was established to study this approach has recently concluded, without any visible impact on the CDN industry, and the amount of CDI research that exists in current literature is also very limited. The ODN concept proposed in this thesis is another feasible approach for cost-effective CDN deployment, which would also be complementary to CDI if it is to be adopted in the future.
- Existing CDN replica placement models are highly simplified. Some placement models only look at server locations and assume that each server could store any number of content items and serve any number of customers. Other models, while taking into account server capacity constraints, only consider capacities in the form of content storage space or replication bandwidth, not server load. As server load is related to the number of requests to be routed to each server, its consideration would significantly increase the complexity of the placement problem compared to existing models. However, we believe it would be a more realistic constraint, considering that the main driving force behind CDN development was server and network overloading, not storage capacity problems.
- In current CDN literature, the majority of studies focus on traditional web content. Only a very limited number of studies have considered CDNs that deliver multimedia content. However, to our knowledge the provisioning problems for CDNs that deliver pay-per-view and live streaming content, which we intend to examine in this thesis, are still not yet addressed.

- Existing CDN studies have mainly concentrated on the last mile issues: the delivery of content from CDN servers to clients. The requirements and support for networking among CDN servers have not been examined. Although this might not be a significant issue in current generation CDNs, we believe it would play an important part in future content applications where transmission paths with quality of service guarantees are required for content distribution among servers, for example in the distribution of live streaming content. In this thesis we intend to study this issue and consider the introduction of extra networking support that enhances such communication in the ODN backbone.
- CDN content replica placement has been addressed as a stand-alone problem with the objective of finding the placement pattern that gives the best performance possible within certain known resource limits. Replica placement in ODN, however, is substantially different and more difficult due to the nature of the shared infrastructure environment. As will be demonstrated in the next chapter, this would render the existing CDN replica placement algorithms unsuitable for ODNs.

Chapter 3

An ODN Provisioning Framework

3.1 Introduction

A major portion of work carried out in this thesis is in developing and studying ODN provisioning models. Before going into details with application-specific provisioning scenarios in the next few chapters, this chapter aims to study the general characteristics and requirements of ODN provisioning and develop a suitable provisioning framework. This framework would specify various aspects of the provisioning process such as objectives, constraints, required input and expected output, the interaction and roles of entities involved, and would be used as a guideline for developing ODN provisioning models in the rest of this thesis.

This chapter is organized as follows. In Section 3.2, the proposed ODN concept will be introduced and described in detail. In Section 3.3, the ODN provisioning problem will be introduced and analyzed. This section will also discuss the differences between provisioning problems in ODNs and traditional CDNs. Based on this analysis, a general ODN provisioning framework will then be developed and described in Section 3.4. Finally, major conclusions from this chapter will be presented in Section 3.5.

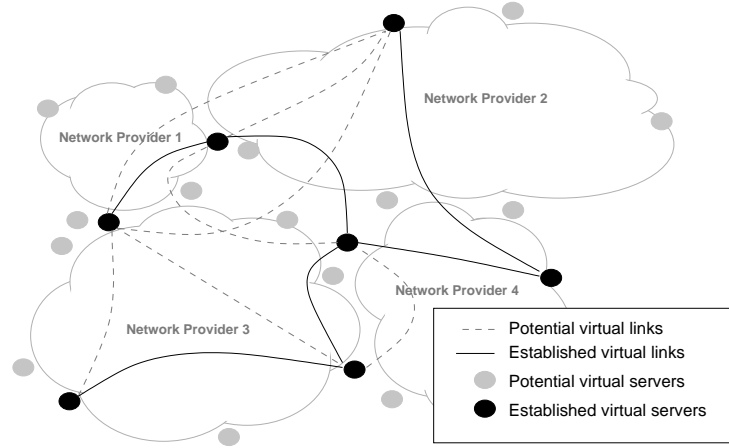


Figure 3.1 An Example ODN Topology

3.2 ODN Concept

With the Overlay Distribution Networks (ODN) concept, we aim to find an alternative, more cost effective and more flexible way of obtaining the necessary server and network resources required by CDNs. In today's CDNs, although the servers are placed in different networks under different administrative domains, they are owned, deployed, maintained and operated by the CDN owner. In contrast, with the ODN approach we propose to establish content distribution networks as overlay networks of leased virtual servers and virtual links. This deployment approach is expected to present less demanding infrastructure investment and resource commitment compared to existing CDNs. It would also create the possibility of adjusting server capacity and even the whole ODN topology on-the-fly in response to fluctuations in demand.

In an ODN, the virtual servers are created by leasing resources from physical servers or server farms, and are used for content replication, delivery and potentially content enhancement functionalities such as adaptation, personalization or localization. The virtual links, on the other hand, are created among servers by leasing transport resources from network providers. They form the backbone of the ODN and are used for intra-ODN traffic such as content replication, update, back-end server communication and other ODN control and management traffic. An example can be seen in

Figure 3.1, where an ODN topology is established on top of many different physical servers and networks.

In this environment, we define the following entities:

- *Server Providers:* These are the owners of physical servers or server farms, who offer logical partitions of their servers for lease in the form of virtual servers. The underlying technologies used to create virtual servers could be different from one server provider to the next, including VMWare, Xen or user-mode Linux processes. We only assume that the partitioning techniques in use are able to create virtual servers with specified amounts of resources, such as processing power, storage space and Internet bandwidth.
- *Network Providers:* These are the owners of physical networks, from whom transport resources could be leased in the form of virtual links between server locations. We also do not make any assumptions on the underlying technologies used to create such virtual links, which could be MPLS label switched paths or other layer 2/3 encapsulation tunnels.

Note that network providers and server providers are collectively referred to as *infrastructure providers* in this thesis.

- *Application Provider:* In the shared infrastructure environment envisaged in this thesis, an ODN is just one of many applications that could co-exist over the same physical infrastructure. For example, over the physical servers and networks available in Figure 3.1, several ODN topologies could be leased and operated by different businesses targeting different user and content markets. Similarly other virtual topologies could also be leased to support applications unrelated to content distribution, such as online games, virtual environments or other distributed computation applications at the same time.

As a result, the term *application provider* is used in this thesis to refer to the owner of a virtual network topology, which in majority of cases means an ODN owner unless indicated otherwise.

- *Resource Brokers:* As the trend of resource virtualization and sharing becomes popular, the number of infrastructure providers offering server and network

resources is likely to be large. Thus there could be *resource brokers* who facilitate the resource leasing process by being intermediaries between infrastructure providers and application providers. However, the existence of such entities would not affect the nature of the ODN provisioning process, which we aim to study in this chapter.

For the ODN concept to become a reality, there must also be a form of resource market that allows the above entities to interact. The interactions could include gathering information on available resources, establishing and leasing virtual servers and virtual links with specified capacities. In our work we will assume the existence of such a market. The implementation of this mechanism, however, is out of the scope of this thesis.

3.3 ODN Provisioning versus CDN Provisioning

Having presented an overview of the ODN concept and its shared infrastructure environment in the previous section, we now aim to examine how this new environment has transformed the ODN provisioning process in comparison with traditional CDNs. The ODN provisioning process aims to determine how the ODN topology should be created. As mentioned earlier in Chapter 1, it consists of the following subtasks:

- *Topology Planning*: This aims to determine the desired ODN topology, including the number of virtual servers and virtual links to be leased and their locations.
- *Resource Dimensioning*: This aims to determine the amount of processing power, storage space and network bandwidth to be leased at each server and for each link.
- *Replica Placement*: This aims to determine in which servers each content item (e.g. web pages, audio or video files) should be replicated and how end users' requests are directed to these replicas.

Tackling these issues appropriately is of paramount importance, as they directly influence the resource efficiency and quality of service that the ODN delivers. For example, as will be seen later in Chapter 4, provisioning ODNs using an inappropriately designed algorithm, which is a simple adaptation of existing CDN replica placement models, could lead to topologies that cost the application provider over 2 times more than what could be achieved with more suitable provisioning algorithms. As reviewed earlier in Chapter 2, replica placement and topology planning for traditional CDNs have also been previously studied in literature. However, the shared infrastructure nature of the ODN approach would significantly transform these problems in ODNs, making them substantially different from and more complex than traditional CDNs. This is due to the following reasons:

Firstly, ODN provisioning process has to take into account many more factors than what have been considered in existing CDN provisioning studies. For example the number of potential infrastructure providers could be large, with costs of individual resources such as processing power, storage space and bandwidth varying from one provider to another or even from one location to another. On top of that, there could also be limits on the maximum amount of each type of resources that could be leased. All these factors affect the desirability of leasing virtual servers and virtual links at each particular location and need to be considered by the provisioning process in order to make cost-effective provisioning decisions.

Secondly, replica placement can no longer be addressed as a stand-alone problem as seen in CDN literature. The required ODN topology and associated resources will influence and are influenced by how content items are replicated and how requests are redirected to these replicas. Therefore, in ODN provisioning, topology planning, resource dimensioning and replica placement are interconnected and will have to be addressed jointly.

Thirdly, the objective of the replica placement process is also different in ODNs. In current CDN literature, replica placement models share a common objective of finding the placement pattern that gives the best performance possible, given certain resource limits. This may be suitable for traditional CDNs, as the CDN provider owns the infrastructure and could use up what has already been deployed in search of the best performance. In ODNs, however, deploying more servers or placing more

content replicas could improve service performance, but will also increase total resource cost, thus creating a cost-performance trade-off. Aiming for the best possible performance in this environment would not be realistic as this may mean creating servers wherever possible, replicating content wherever possible and leasing as much resources as possible, at a huge cost. Instead, the optimal ODN topology would be the one that delivers satisfactory service performance at the minimum cost.

3.4 ODN Provisioning Framework

The goal of the ODN provisioning process is to determine the optimal ODN topology, including locations of virtual servers and virtual links, the amount of processing power, storage space and bandwidth to be allocated at each location. In order to do this, it will also have to determine where each content item is to be replicated and how content requests are to be rerouted, as these parameters and resource requirements are interdependent. In other words, the ODN topology planning, resource dimensioning and replica placement problems are to be addressed jointly.

As pointed out in the previous section, the optimal ODN topology would be the one that delivers satisfactory service performance at the minimum cost. Therefore, at the heart of the ODN provisioning process would be an optimization problem. This optimization aims to find the optimal ODN topology, content replication and request routing parameters, given available infrastructure choices and estimated application requirements. In particular, the following information would be taken into account:

- *Available resources:* These include potential server and link locations, their various resource costs and capacity limits. This information comes from infrastructure providers who have server and network resources available for lease.
- *Application requirements:* These include content items to be distributed and their resource requirements, user locations and estimated demand levels, and the service performance to be achieved. This information comes from the application provider, who leases and uses the ODN to deliver content.

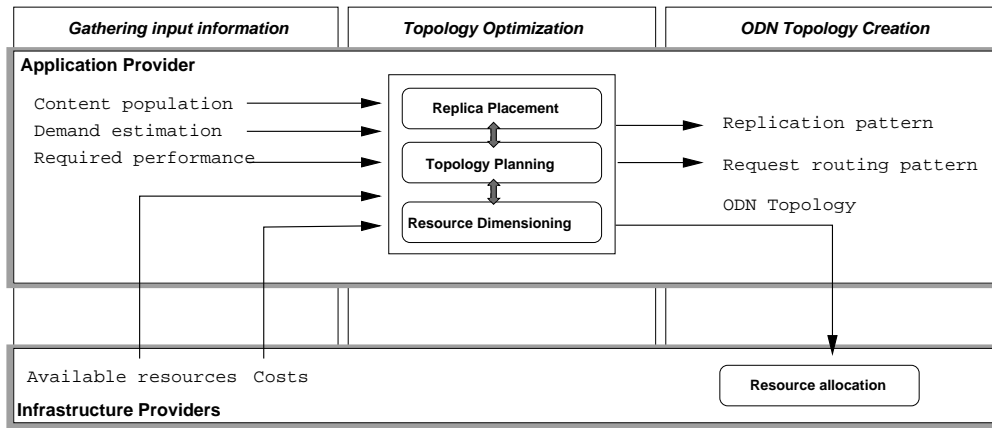


Figure 3.2 ODN Provisioning Framework

The input information required by the ODN provisioning process therefore would need to be collected from both the application provider and potential infrastructure providers (or resource brokers representing these providers). The provisioning process, however, would be carried out by the application provider. This is because the optimization problem that looks for the optimal ODN topology is highly dependent on the characteristics of the application to be delivered through the ODN. For example, an ODN that distributes web content and an ODN that distributes high quality live streaming media would naturally need very different topology optimization models. As a result, the responsibility for carrying out this optimization would rest with the application provider, who has the most intimate knowledge of the application's operation and resource requirements. Infrastructure providers or resource brokers, on the other hand, would only participate by providing the required information on available resources and establishing the resulting topology when requested.

Therefore we envisage that each application provider would use a provisioning tool that implements an application-specific topology and resource optimization model. This provisioning tool would: (1) gather the necessary information from infrastructure providers or resource brokers, (2) compute the desired topology, and (3) pass this information back to infrastructure providers, who will then create the ODN topology and allocate the requested resources. By periodically repeating this optimization process at appropriate time intervals, it would also be possible for a service provider to

adjust the ODN's resource usage in order to reduce cost or cope with fluctuations in service demand.

The steps involved in this provisioning process, as well as the flow of input and output information, are illustrated by the diagram in Figure 3.2.

3.5 Conclusion

In this chapter, we have introduced the ODN concept and examined the ODN provisioning problem. We have analyzed the characteristics and requirements of this provisioning problem and pinpointed the differences between provisioning in ODNs and traditional CDNs.

In summary, due to the shared infrastructure environment, the ODN provisioning problem would be significantly different from and more complex than what have been considered in current CDN literature. ODN provisioning would involve addressing topology planning, resource dimensioning and replica placement problems jointly instead of as stand-alone problems seen in CDN literature. ODN provisioning would also have a different objective. It would aim to find the topology and replica placement pattern that gives satisfactory performance at the minimum cost rather than striving for the best performance possible as in existing CDN replica placement models.

Based on these observations, a suitable provisioning framework has been developed that describes how the provisioning process would be done for ODNs. According to this framework, ODN provisioning would involve an optimization process that determines the optimal ODN topology and other application-specific parameters such as content replication and request routing, which deliver satisfactory performance at the lowest total resource cost. This optimization would require information from both infrastructure providers and the application provider who is establishing the ODN, and is expected to be performed by the application provider.

Using this general provisioning framework, in the next few chapters of this thesis we will examine and develop ODN provisioning models for a number of different applications, including web content, pay-per-view content and live streaming multimedia content distribution.

Chapter 4

Provisioning ODN for Web Content Distribution

4.1 Introduction

This chapter considers the provisioning problem of ODNs established for web content distribution applications. Assuming that the costs, locations and available resources of potential servers and network links are provided, the set of content items to be distributed are given and the location and request rates of user population could be estimated, this problem aims to find the optimal ODN configuration to be used. This includes determining the overlay network topology, the required amount of resources, the replication locations of each content item and the routing of user requests.

Although a large number of studies in current CDN research literature have addressed content replica placement and topology planning issues (Kangasharju et al., 2001a; Venkataramani et al., 2001; Qiu et al., 2001), the ODN provisioning problem considered in this chapter is significantly different. As pointed out in previous chapters, due to the “resource leasing” nature of the ODN approach, ODN provisioning would involve not only interdependent topology planning, resource dimensioning and replica placement problems, but also a different optimization objective. Instead of searching for the best performance possible, ODN provisioning aims to find an ODN configuration that delivers satisfactory performance at the minimum cost.

The problem will be formulated in this chapter as a mixed integer linear program-

ming model, with the assumptions that application performance is measured by averaged user-content distances, and that web pages are typically small in size and require similar amounts of processing and bandwidth per content request. In these aspects our formulation is similar to existing models in CDN content replica placement literature. However, while in existing models servers have either no or only storage-based capacities and user requests are simply directed to the nearest replica, we try to avoid such oversimplification. We aim to take into account request routing and model server capacities based on the amount of requests a server can handle. We believe this is important as the ability to spread demand and avoid server overloading is one of the key features that make the CDN solution attractive in the first place. As this ODN provisioning problem belongs to the class of NP problems, we will then develop heuristic procedures that allows near optimal solutions to be found efficiently.

This chapter is organized as follows. In Section 4.2 the variables and parameters used to formulate the ODN provisioning problem will be described and the problem formulated as a mixed integer linear programming optimization. The optimization, which has both binary and continuous variables, can be proven to be an NP problem, as demonstrated in Section 4.2.4. Therefore two heuristic procedures will then be proposed and developed to find its near optimal solutions, one based on Lagrangian relaxation and subgradient optimization (Section 4.3) and the other a two-level greedy search algorithm (Section 4.4). As a comparison benchmark, another greedy heuristic designed based on the principles of existing CDN replica placement heuristics is also developed, which will demonstrate that significant performance gains are achieved by the heuristics that are based on the formulation structure. Numerical results, obtained with C++ implementation of these heuristics, will be presented and analyzed in Section 4.6. Finally, concluding remarks derived from the studies in this chapter will be presented in Section 4.7.

4.2 Provisioning Problem Formulation

4.2.1 Input Parameters

The input information required by the ODN provisioning process includes potential resources and costs, content items to be distributed, user locations, demand estimates and the performance level to be achieved.

The problem is modelled using a graph $G = \langle V, E \rangle$, where V is the set of nodes and E is the set of edges. Two subsets of nodes are defined - access nodes V_a and potential server nodes V_s , where $V = V_a \cup V_s$, $N_a = |V_a|$ and $N_s = |V_s|$. An *access node* represents the location of end users and the source of contents requests. Request rates are assumed to be estimated per access node instead of individual users. In real life, such an access node could be an ISP access network or point-of-presence.

A potential *server node* represents a location that processing power, storage space and Internet bandwidth can be leased and a virtual server can be established, such as a server farm or server cluster. To model server capacities, each potential server $i \in V_s$ has a limit C_i on the amount of requests it can handle per unit time.

Edges in E represent network links and each edge $e \in E$ has a weight that reflects the network distance between two nodes. The distance τ_{im} between two nodes $i \in V_s$ and $m \in V_a$ is used to measure the access latency when content is delivered from server i to access node m . This distance is pre-calculated using shortest path algorithms on graph G . The performance of the ODN is measured as the average of these content access latencies.

We also define a content population of K items (e.g. web pages, images or whole web sites), with item $k \in 1, 2, \dots, K$ having size α_k . The amount of requests generated by customer node m for item k per unit time is represented by λ_{mk} .

Establishing a virtual server is assumed to incur a cost for initiating and maintaining the server, plus costs for the various virtual resources such as storage, computation and bandwidth. For each server node i , these parameters are denoted as follows:

- c_i : server cost, which models the costs involved in establishing and maintaining a virtual server (e.g. resource allocation, shipping software, configuration...)

- d_i : unit storage cost
- a_i : unit access cost, which represents the bandwidth and processing power cost due to each content request.
- C_i : maximum server capacity

As the web content being distributed through the ODN is likely to come from content providers that buy the service from the ODN owner, it is assumed that the ODN owner has the obligation to deliver content items to end users with a certain level of performance, which is modeled by a maximum “acceptable threshold” T_{max} in access latency.

4.2.2 Decision Variables

To formulate the ODN provisioning problem, a number of binary and continuous decision variables are defined to represent the configuration parameters of the ODN topology. These variables include:

$$y_i = \begin{cases} 1 & \text{if site } i \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

$$x_{ik} = \begin{cases} 1 & \text{if item } k \text{ is replicated at site } i \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

$$r_{mi}^k \in [0, \lambda_{mk}] \quad (4.3)$$

with the continuous variables r_{mi}^k used to indicate the fraction of requests for item k from customer m that should be directed to server i .

Through these variables the amount of resources to be leased at each virtual server can be calculated as follows:

- storage capacity = $\sum_{k \in K} x_{ik} \alpha_k$, and
- delivery capacity = $\sum_{m \in V_a} \sum_{k \in K} r_{mi}^k$, for server $i \in V_s$

For convenient reference, a complete list of the above parameters and decision variables can be found in Table 4.1.

The aim of the provisioning problem now is to find the optimal ODN configuration through determining the optimal values of the decision variables. This can be done by solving the optimization model in the following section.

Table 4.1 Parameters and Variables for Web Content ODN Provisioning Model

Notation	Meaning
<i>Resource parameters</i>	
a_i	unit access cost (per content request) at server i
c_i	server cost
d_i	server unit storage cost
C_i	server delivery capacity
τ_{im}	distance between server i and access node m
N_s	number of server nodes, $N_s = V_s $
N_a	number of access nodes, $N_a = V_a $
i	subscript for server nodes
m	subscript for access nodes
<i>Content parameters</i>	
α_k	size of content item k
λ_{mk}	request rate for item k from access node m
k	subscript for content items
<i>Decision variables</i>	
y_i	server establishment indicator, $y_i \in 0, 1$
x_{ik}	content replication indicator, $x_{ik} \in 0, 1$
r_{mi}^k	request redirection ratio, $r_{mi}^k \in [0, \lambda_{mk}]$

4.2.3 Optimization Model

Using the parameters and decision variables defined above, the following optimization model formulates the ODN provisioning problem by aiming to deliver all content items within the agreed access latency at the minimum total cost.

$$(P) \quad \min \underbrace{\sum_{i \in V_s} \sum_{k \in K} x_{ik} d_i \alpha_k}_{\text{Storage cost}} + \underbrace{\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k a_i}_{\text{Delivery cost}} + \underbrace{\sum_{i \in V_s} y_i c_i}_{\text{Server cost}} \quad (4.4)$$

Subject to:

$$\sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \leq C_i, \forall i \in V_s \quad (4.5a)$$

$$\sum_{i \in V_s} r_{mi}^k = \lambda_{mk}, \forall m \in V_a, k \in K \quad (4.5b)$$

$$\frac{\sum_{i \in V_s} \sum_{m \in V_a} r_{mi}^k \tau_{im}}{\sum_{m \in V_a} \lambda_{mk}} \leq T_{max}, \forall k \quad (4.5c)$$

$$\sum_{i \in V_s} y_i C_i \geq \Lambda \quad (4.5d)$$

$$r_{mi}^k \leq \lambda_{mk} x_{ik}, \forall m, i, k \quad (4.5e)$$

$$x_{ik} \leq y_i, \forall i, k \quad (4.5f)$$

$$x_{ik}, y_i \in \{0, 1\} \quad (4.5g)$$

$$r_{mi}^k \in [0, \lambda_{mk}] \quad (4.5h)$$

where $\Lambda = \sum_{m \in V_a} \sum_{k \in K} \lambda_{mk}$ is the total demand from all customers. In this formulation, the objective is to minimize the sum of server and resource costs, while maintaining acceptable performance. Constraint (4.5a) is used to enforce the capacity limit at each potential server site. Constraint (4.5b) means that all requests must be served, while constraint (4.5c) maintains the average access latency threshold for each item.

The other constraints simply establish the logical relationships among different variables. For example constraint (4.5e) means that a site can only serve request for an item if that item is replicated there, and (4.5f) means a site must be in use for items to be replicated there.

Constraint (4.5d) ensures that the total delivery capacity of opened sites must not be smaller than the global sum of request rates. Note that this is actually a redundant constraint, which can be derived from (4.5a), (4.5e) and (4.5f). Although redundant, it has been added to strengthen a Lagrangian relaxation approach that we will be using in Section 4.3. Similar techniques have often been used in operations research literature (Sridharan, 1995).

Also note that in this model we have ignored the cost of inter-server virtual links. This is because, similar to existing works in CDN content replica and server place-

ment, we have assumed that the web content items are relatively static, and thus the amount of bandwidth consumed by delivering web content to end users from ODN servers would be significantly higher than the backbone bandwidth consumed by replicating and updating them. As a result, a simplified approach is used where only the more significant factors - ODN server locations and their resources, including storage, computation power and delivery bandwidth - are determined in this optimization model. The less important inter-server virtual links could then be provisioned based on established server locations, in a separate subprocess, for example with one of the link provisioning models to be developed in Chapter 8.

4.2.4 Problem Complexity

In this section we examine the complexity of the optimization model developed above and aim to show that it belongs to the NP class of problems.

The number of variables and constraints in the ODN provisioning formulation are:

- Binary variables: $N_s + N_s K$, i.e. $O(N_s K)$
 Continuous variables: $N_a N_s K$
 Total number of variables: $N_s + N_s K + N_a N_s K$, i.e. $O(N_a N_s K)$
- Constraints: $N_s + N_a K + 2 + N_a N_s K + N_s K$, i.e. $O(N_a N_s K)$

The problem can be proven to be NP-hard by reducing it to the well known NP-hard facility location problem (Sridharan, 1995) as follows:

Consider the special scenario where $K = 1$, $c_i = 0, \forall i$, $T_{max} = \infty$ and denote $v_{im} = \frac{r_{mi}}{\lambda_m}$, $v_{im} \in [0, 1]$. In this case, the subscript for k can be dropped since there is only one item and the start up cost and performance constraint (4.5c) can be eliminated. The problem therefore can be rewritten as:

$$\text{Min} \sum_{i \in V_s} \sum_{m \in V_a} v_{im} \lambda_m a_i + \sum_{i \in V_s} x_i d_i \alpha$$

Subject to:

$$\begin{aligned}
\sum_{m \in V_a} v_{im} \lambda_m &\leq C_i, \forall i \\
\sum_{i \in V_s} v_{im} &= 1, \forall m \\
0 &\leq v_{im} \leq x_i \leq 1, \forall i, m \\
x_i &\in \{0, 1\}
\end{aligned}$$

which is in the form of a capacitated facility location problem. As a result, the provisioning model developed above is also an NP-hard problem.

4.3 Lagrangian Heuristic

As the ODN provisioning problem formulated above is an NP-hard problem, we may not be able to find its optimal solution efficiently, especially at large problem sizes. Instead we will develop heuristic procedures that aim to find near optimal solutions with reasonable computation effort.

The heuristic developed in this section is based on Lagrangian relaxation, which is a popular technique in solving complex optimization problems (Glover and Laguna, 1993). In this heuristic, instead of solving the original difficult problem, we will iteratively solve a series of simpler ones that are generated by relaxing a number of constraints. At each iteration, the solution to the relaxed problem may or may not be feasible for the original problem, but is used as a basis from which a candidate feasible solution is constructed. Due to the properties of the Lagrangian relaxation technique, solving the relaxed problem also gives a bound on the original problem's optimal solution, which can be used to assure quality of the candidate solutions. A subgradient procedure is used to steer the iterative process so that candidate solution quality is gradually improved.

4.3.1 Relaxing the Problem

Using Lagrangian relaxation techniques, optimization models can be simplified by relaxing constraints and including them in the objective function. In this case, constraint (4.5e) was chosen to be relaxed as its elimination would allow the problems to be split into independent subproblems that could be solved efficiently. The constraint is relaxed and the following term is added to the objective function:

$$\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} \mu_{mi}^k (r_{mi}^k - \lambda_{mk} x_{ik}), \mu_{mi}^k \geq 0$$

where μ_{mi}^k is the Lagrangian multiplier. The objective function now becomes:

$$\begin{aligned} \sum_{i \in V_s} \sum_{k \in K} x_{ik} d_i \alpha_k + \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k a_i + \sum_{i \in V_s} y_i c_i + \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} \mu_{mi}^k (r_{mi}^k - \lambda_{mk} x_{ik}) \\ = \left[\sum_{i \in V_s} \sum_{k \in K} x_{ik} (d_i \alpha_k - \sum_{m \in V_a} \lambda_{mk} \mu_{mi}^k) \right] + \sum_{i \in V_s} y_i c_i \\ + \left[\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (a_i + \mu_{mi}^k) \right] \end{aligned} \quad (4.6)$$

By rearranging the relaxed objective function and remaining constraints, the original problem can be decomposed into two following subproblems:

$$(P1) \quad \text{Min} \sum_{i \in V_s} \sum_{k \in K} x_{ik} (d_i \alpha_k - \sum_{m \in V_a} \lambda_{mk} \mu_{mi}^k) + \sum_{i \in V_s} y_i c_i \quad (4.7)$$

Subject to:

$$\sum_{i \in V_s} y_i c_i \geq \Lambda \quad (4.8a)$$

$$x_{ik} \leq y_i, \forall i, k \quad (4.8b)$$

$$x_{ik}, y_i \in \{0, 1\} \quad (4.8c)$$

and

$$(P2) \quad \text{Min} \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (a_i + \mu_{mi}^k) \quad (4.9)$$

Subject to:

$$\sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \leq C_i, \forall i \in V_s \quad (4.10a)$$

$$\sum_{i \in V_s} r_{mi}^k = \lambda_{mk}, \forall m \in V_a, k \in K \quad (4.10b)$$

$$\frac{\sum_{i \in V_s} \sum_{m \in V_a} r_{mi}^k T_{im}}{\sum_{m \in V_a} \lambda_{mk}} \leq T_{max}, \forall k \quad (4.10c)$$

$$r_{mi}^k \in [0, \lambda_{mk}] \quad (4.10d)$$

Note that all binary variables have been separated into problem $(P1)$, and the continuous variables into $(P2)$. For each value of the Lagrangian multiplier vector μ , the relaxed problem can be solved by solving $(P1)$ and $(P2)$ separately, as described in the following subsection.

4.3.2 Solving Subproblems

To solve subproblem $(P1)$, constraint (4.8a) is first left aside. The resulting problem is then further decomposed into N_s smaller problems, one for each location i :

$$(P1_i) \quad \text{Min} \sum_{k \in K} x_{ik} \left(d_i \alpha_k - \underbrace{\sum_{m \in V_a} \lambda_{mk} \mu_{mi}^k}_{F_{ik}} \right) + y_i c_i \quad (4.11)$$

Subject to:

$$x_{ik} \leq y_i, \forall i, k \quad (4.12a)$$

$$x_{ik}, y_i \in \{0, 1\} \quad (4.12b)$$

For each of these N_s problems, there are the following possibilities:

1. if $y_i = 0$ (server location i unused) then:

$$x_{ik} = 0, \forall k \in 1, 2, \dots, K, \text{ due to constraint (4.12a).}$$

2. if $y_i = 1$ (server location i used) then:

- for each $k \in K$
 - if $F_{ik} \geq 0$ then $x_{ik} = 0$

- if $F_{ik} < 0$ then $x_{ik} = 1$

where F_{ik} is defined in (4.11).

- Thus if server location i is used, the resulting cost would be $C(P1_i) = \sum_{k=1}^K x_{ik} F_{ik} + \gamma_i$, where the variables x_{ik} take values determined in above.

Server i therefore would incur a cost of $C(P1_i)$ if used, and 0 otherwise. Therefore if $P1_i \leq 0$, server i should be opened in the optimal solution to problem $(P1)$, i.e. $y_i = 1, \forall i \in V_s^- = \{i | P1_i \leq 0\}$.

If $\sum_{i \in V_s^-} C_i \geq \Lambda$, then constraint (4.8a) is satisfied, and V_s^- is optimal set of servers to be opened. Otherwise it can then be enforced by solving the following problem, which determines which additional servers should also be opened to provide enough delivery capacity.

$$\text{Min} \quad \sum_{i \in V_s \setminus V_s^-} y_i C(P1_i)$$

Subject to:

$$\sum_{i \in V_s \setminus V_s^-} y_i C_i \geq \Lambda - \sum_{i \in V_s^-} C_i$$

which is a binary knapsack problem of at most size N . There exist exact algorithms to solve this problem, for example dynamic programming algorithms found in (Martello and Toth, 1990), (Nemhauser and Wolsey, 1999).

Although the addition of constraint (4.5d) has made solving this subproblem more complex than without it, results from our experiments showed that its inclusion does lead to improvement in solution quality.

Problem $(P2)$, on the other hand, is a constrained multi-commodity minimum cost flow problem and is a pure linear programming problem. It can be solved using standard linear programming solution methods. In this work, the Cplex optimization package (Cplex, 2001) is used to solve this problem.

Regardless of the values of multiplier μ , the optimal objective function of the relaxed problem is always a lower bound of the optimal objective of the original problem (P) , due to the properties of Lagrangian relaxation:

$$\text{Obj}^*(P_{\text{relaxed}}(\mu)) = \text{Obj}^*(P1(\mu)) + \text{Obj}^*(P2(\mu)) \leq \text{Obj}^*(P) \quad (4.13)$$

where $Obj^*(.)$ denotes the optimal objective function value of an optimization problem.

4.3.3 Constructing a Feasible Solution

As $(P1)$ and $(P2)$ were created by relaxing original problem (P) , the set of variable values (x_{ik}, y_i, r_{mi}^k) obtained by solving the subproblems may have violated the relaxed constraints and thus be an infeasible solution to (P) .

We therefore need to construct candidate solutions that are feasible to the original problem. In this heuristic this is done by accepting request routing parameters (r_{mi}^k) as determined in subproblem $(P2)$ and make content replica and server placements necessary to make this request routing pattern feasible. In other words, the following procedure is used:

- Set $\bar{x}_{ik} = 0$ and $\bar{y}_i = 0, \forall i, k$
- Consider the solution $\{r_{ij}^k\}$ obtained by solving subproblem $P2$
- For each r_{ij}^k
set \bar{x}_{ik} and \bar{y}_i to 1 if $r_{ij}^k > 0$

It can be easily proven that the set $(r_{ij}^k, \bar{x}_{ik}, \bar{y}_i)$ created by this procedure is a feasible solution to the original model. Obviously the resulting objective function value cannot be better (smaller) than the optimal objective:

$$Obj(P(r_{ij}^k, \bar{x}_{ik}, \bar{y}_i)) \geq Obj^*(P) \quad (4.14)$$

In other words, each constructed candidate solution gives an upper bound on the optimal objective function. The gap between this upper bound and the lower bound in (4.13) can be used to measure the quality of a candidate solution, as it is the worst case distance between the candidate solution and the optimal one.

4.3.4 Subgradient Procedure

The heuristic iteratively constructs a series of candidate solutions by solving a series of relaxed problems with different Lagrangian multiplier values (μ) . The popular

subgradient method (Glover and Laguna, 1993) is used to adjust the multiplier values between iterations in a way to improve the gap between upper and lower bounds, and thus improve the quality of candidate solutions. The procedure stops after a large enough number of iterations have passed or a good enough candidate solution has been found.

Details of the subgradient procedure used in this heuristic is presented in Appendix A.2, and further background information on this method, which is commonly used in developing Lagrangian heuristics, can be found in (Glover and Laguna, 1993).

4.4 Two-Level Greedy Search Heuristic

The second heuristic developed for this web ODN provisioning problem involves a two-level greedy search. In this heuristic a greedy search is performed to find candidate ODN topologies. For each candidate topology another greedy search is then carried out to find suitable replica placement and request routing patterns. The development of this heuristic procedure is described below.

4.4.1 Analysis

In the original optimization (4.4), if values of variables y_i were determined, i.e. if the topology were fixed, the objective function would become:

$$(P_T) \text{ Min } \sum_{i \in V_s^+} \sum_{k \in K} x_{ik} d_i \alpha_k + \sum_{i \in V_s^+} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k a_i + \sum_{i \in V_s^+} c_i \quad (4.15)$$

where V_s^+ is the set of open servers, whose total start up costs $\sum_{i \in V_s^+} c_i$ is already fixed and can be removed from the optimization objective. Note that the objective function now can be broken up across k (items), and that in the constraint set, only constraint (4.5a) binds the items together. Thus we can approximate (P_T) by ranking the items (e.g. according to total demand or priority) and solving the following problem for each item sequentially. The resulting problem for item k is below, with subscript k dropped for convenience.

$$(P_{Tk}) \text{ Min } \sum_{i \in V_s^+} x_i d_i \alpha + \sum_{i \in V_s^+} \sum_{m \in V_a} r_{mi} a_i \quad (4.16)$$

Subject to:

$$\sum_{m \in V_a} r_{mi} \leq C_i^{residual,k}, \forall i \quad (4.17a)$$

$$\sum_{i \in V_s^+} r_{mi} = \lambda_m, \forall m \quad (4.17b)$$

$$\frac{\sum_{i \in V_s^+} \sum_{m \in V_a} r_{mi} \tau_{im}}{\sum_{m \in V_a} \lambda_m} \leq T_{max}, \quad (4.17c)$$

$$r_{mi} \leq \lambda_m x_i, \forall i, m \quad (4.17d)$$

$$x_i \in \{0, 1\}; r_{mi} \in [0, \lambda_m] \quad (4.17e)$$

Note that the residual resource has to be updated after the optimization is done for each item: $C_i^{residual,k} = C_i^{residual,k-1} - \sum_{m \in V_a} r_{mi}^{k-1}$.

(P_{Tk}) is similar to a facility location problem (FLP) with an extra performance constraint. However, if the set x_i (i.e. item replication) were fixed, the problem could be further reduced the following form, which is a constrained transport problem:

$$(P_{trans}) \quad \sum_{i \in V_s^{x+}} \sum_{m \in V_a} r_{mi} a_i \quad (4.18)$$

Subject to:

$$\sum_{m \in V_a} r_{mi} \leq C_i^{residual,k}, \forall i \quad (4.19a)$$

$$\sum_{i \in V_s^{x+}} r_{mi} = \lambda_m, \forall m \quad (4.19b)$$

$$\frac{\sum_{i \in V_s^{x+}} \sum_{m \in V_a} r_{mi} \tau_{im}}{\sum_{m \in V_a} \lambda_m} \leq T_{max}, \quad (4.19c)$$

$$r_{mi} \leq \lambda_m, \forall i, m \quad (4.19d)$$

$$r_{mi} \in [0, \lambda_m] \quad (4.19e)$$

where V_s^{x+} is the set of servers that contain replicas of the item in question. In this reduced form the problem can be solved efficiently to find the optimal values of variables r_{mi} .

4.4.2 Heuristic Outline

Based on the above observations, a two-level greedy search heuristic has been developed. The first level of this heuristic attempts to search for candidate topologies and

the second level searches for the best replication pattern within each topology. The greedy search at each level is based on a classic drop procedure (Sridharan, 1995) and is described below:

Topology search:

1. Initialize: Start with all server locations opened, calculate the cost of current topology.
2. For each node i in the current topology, find the total cost of the resulting topology if server at i is removed.
3. If there are possible cost savings, close server i that offers the maximum saving.
4. Go back to (2) and continue till no improvements can be found.

Replica placement search: For each candidate topology, a suitable (i.e. low cost) content replica placement and request routing pattern is found using the following procedure.

1. Rank the items in decreasing order of total demand.
2. For each item k
 - Start by placing a replica of k at every server and calculate current cost by solving a constrained transport problem.
 - For each replica, re-optimize the problem to find the cost saving if it were removed.
 - Drop the replica that offers the maximum saving.
 - Continue drop attempts till no further improvements can be found.
3. Update the residual resource remained at each server and proceed to next item.

Thus instead of solving the original problem, we solve a large number of simpler constrained transport problems of size at most $M.N$. The number of transport problems can be shown to be $O(N^2K)$. In the implementation of this heuristic, optimization package Cplex is used to solve the transport problems.

A drawback of this greedy heuristic is that it does not always guarantee to find a feasible solution, even if one exists. This is because the heuristic effectively performs a series of local searches but do not examine the whole search space, i.e. all possible values of variables (x_{ik}, y_i, r_{mi}^k) . As a result, the heuristic might not be able to find solutions for all feasible problems. This situation would be more likely to happen if available server capacity limits are very close to the required capacities, and less likely otherwise. The Lagrangian heuristic, in contrast, would always find a solution if the problem is feasible, because as long as subproblem $(P2)$ (4.9) is solvable, a feasible solution can always be reconstructed.

4.5 CDN-based Greedy Search Heuristic

Apart from the two heuristics described above, we have also developed another greedy heuristic, which is based on the principles of the greedy replica placement methods used in existing CDN studies by (Kangasharju et al., 2001a; Venkataramani et al., 2001; Qiu et al., 2001). This was developed in order to provide a comparison benchmark for the proposed Lagrangian and two-level greedy search heuristics.

In this greedy heuristic, we start with no servers established and all content requests not handled and then gradually allocate demand from access nodes to server locations until all requests has been allocated. An ODN topology and content replica placement pattern is finally derived from this request rerouting pattern. At each step of this heuristic, we search for the demand allocation that requires minimum cost per request. The steps involved in this heuristic are described in detail as follows:

1. Initialize: The heuristic starts with no servers established, no replicas placed, no content requests allocated to any servers, residual demand for each content item at each access node equal to the original total demand and residual capacity for each server equal to the original capacity. In other words we have:

$$y_i = 0, x_{ik} = 0, r_{mi}^k = 0, \lambda_{mk}^{residual} = \lambda_{mk} \text{ and } C_i^{residual} = C_i, \forall m, i, k$$

2. Searching for the best demand allocation: In this step we consider the possibility of allocating demand for each content item k at each access node m to each server node i . For each of these $\langle m, i, k \rangle$ combinations, we have the following possibilities:

- If the distance to the server is smaller than T_{max} ($\tau_{im} \leq T_{max}$), then the maximum amount of request that could be allocated is:

$$\hat{r}_{mi}^k = \min\{C_i^{residual}, \lambda_{mk}^{residual}\}$$

- If the server is too far away ($\tau_{im} > T_{max}$), some requests could still be allocated, but only if the current average latency of already allocated requests for content item k is smaller than T_{max} . In this case we have:

$$\hat{r}_{mi}^k = \min\{C_i^{residual}, \lambda_{mk}^{residual}, \Psi_k(\frac{T_{max} - \sigma_k}{\tau_{im} - T_{max}})\}$$

where $\Psi_k = \sum_{m,i} r_{mi}^k$ is the total amount of requests that have been allocated for content item k and $\sigma_k = \frac{\sum_{m,i} r_{mi}^k \tau_{im}}{\sum_{m,i} r_{mi}^k}$ is the average access latency of these requests.

We need to enforce $\hat{r}_{mi}^k \leq \Psi_k(\frac{T_{max} - \sigma_k}{\tau_{im} - T_{max}})$ so that the resulting average access latency for item k is still smaller than T_{max} , in order to satisfy performance constraint (4.5c).

If \hat{r}_{mi}^k amount of requests for item k are to be directed from access node m to server i , the resulting additional cost would be:

$$\xi(m, i, k) = a_i \hat{r}_{mi}^k + (1 - y_i) c_i + (1 - x_{ik}) d_i \alpha_k$$

which includes the new access cost, new server cost if server i is not already used, and new storage cost if a replica for item k is not already present at this server.

During this step, the heuristic will look for the allocation that gives the smallest additional cost per request (i.e. $\frac{\xi(m, i, k)}{\hat{r}_{mi}^k}$).

3. Assign requests to server according to the best allocation option found in Step 2 and update the decision variables, residual resources and residual demand levels accordingly:

$$r_{mi}^k = r_{mi}^k + \hat{r}_{mi}^k, y_i = 1, x_{ik} = 1$$

$$\begin{aligned}\lambda_{mk}^{residual} &= \lambda_{mk}^{residual} - \hat{r}_{mi}^k \\ C_i^{residual} &= C_i^{residual} - \hat{r}_{mi}^k\end{aligned}$$

4. The heuristic will stop if all demand has been allocated to servers, otherwise it will go back to Step 2 and attempt to allocate more of the remaining requests.

Note that although this heuristic employs the same design principles of existing CDN replica placement algorithms, significant changes have been introduced. At each iteration the heuristic searches for the best demand allocation instead of content replica locations as in existing studies. This is because in ODN provisioning, server capacity constraints are modelled based on demand, not content storage space. The heuristics also have to take into account more types of resource costs and the fact that the ODN provisioning problem needs to maintain a certain performance level.

Similar to the heuristic described in Section 4.4, a draw back of this CDN-based greedy heuristic is that it might not be able to find solutions to all feasible provisioning problems.

4.6 Numerical Results

The proposed heuristics have been implemented and tested on a number of different network topologies generated by the GT-ITM topology generator (Calvert et al., 1997). In each topology, a number of nodes are randomly selected to be potential server nodes, and the others are assumed to be access nodes. Other parameters, including server capacities, request rates and resource costs are randomly generated. Each topology was tested with different cost combinations in which resource costs are varied across a wide range. All computation was done in a machine with a 1.4GHz CPU and 512MB RAM.

4.6.1 Performance of CDN-based Greedy Heuristic

Comparing the output of the three heuristics, all our experiments indicated that the Lagrangian and two-level greedy heuristics give close performance in terms of solution quality, while the CDN-based greedy heuristic performs consistently and de-

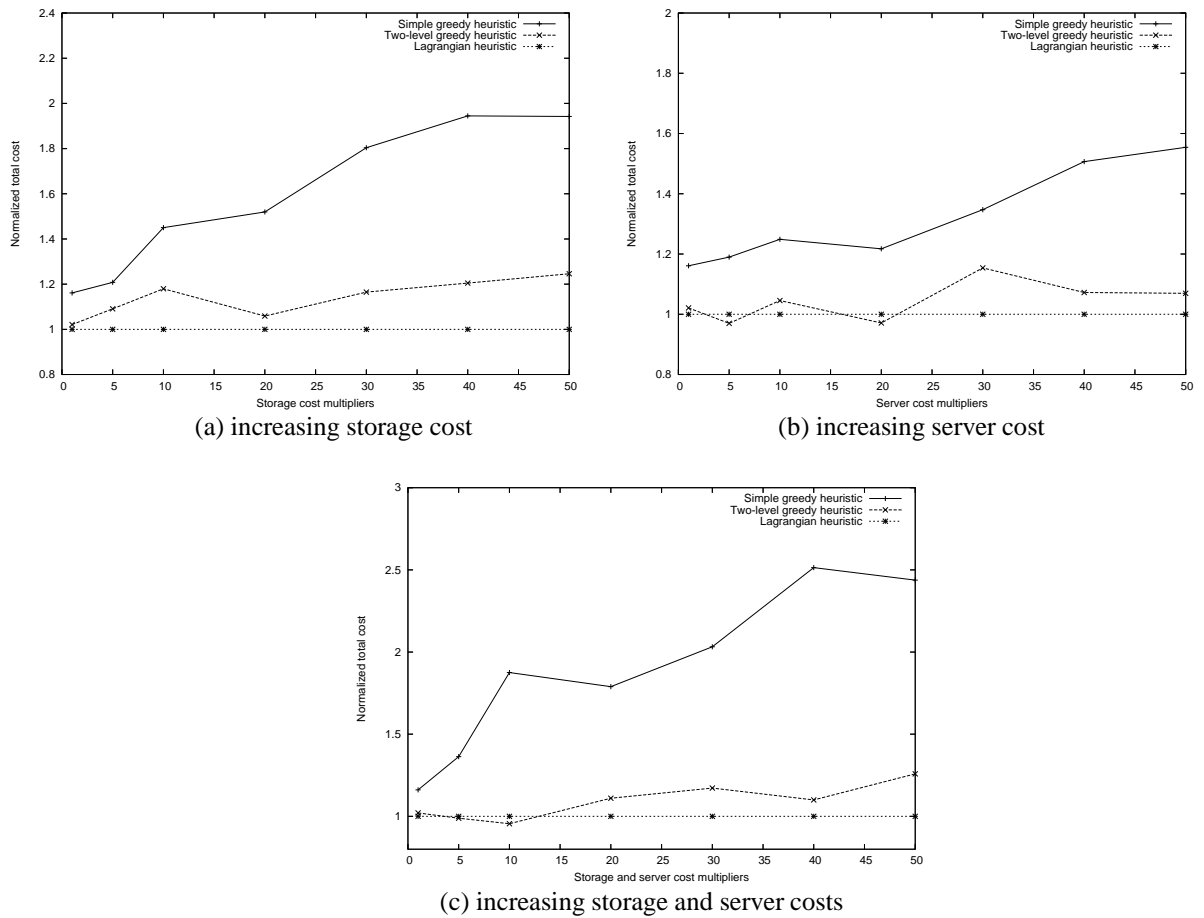


Figure 4.1 Comparing ODN costs by different provisioning heuristics in the presence of different cost variations

cidedly worse. The difference increases as storage and server costs increase, which indicates unnecessary server establishment and content replication.

This can be seen from Figure 4.1, which compares the provisioning costs given by the three heuristics in a series of problems with different resource cost settings. In the experiments shown in Figure 4.1, we start with a base problem where access cost is dominant and accounts for about 90% of the total cost, and gradually increase either storage cost (Figure 4.1.a) or server cost (Figure 4.1.b), or both storage and server costs (Figure 4.1.c) using different multiplier values. Note that in these figures the provisioning costs produced by different heuristics have been normalized by the cost of the Lagrangian relaxation heuristic in order to plot results from different problems in the same graph. The results shown here clearly indicate that significant gains

in solution quality have been achieved by designing heuristic procedures through studying the structure of the problem formulation rather than using an adaptation of existing CDN placement methods. As seen from Figure 4.1.c, the difference could be up to 2.5 times in the total ODN topology cost. Therefore, although the CDN-based heuristic is observed to be significantly faster than the other heuristics, we believe its overall performance is not adequate and will not analyze its output in further details. The trend observed in Figure 4.1, while interesting, is not outside expectation. This is because the CDN-based greedy heuristic makes placement decisions based only on the incremental cost due to individual request allocations, without coordination among different servers or different access nodes. Thus compared to the other heuristics, this algorithm is likely to end up creating more servers and more content replicas than necessary.

4.6.2 Performance of Lagrangian and Two-Level Greedy Heuristics

As illustrated earlier in Figure 4.1, the Lagrangian and two-level greedy heuristics produce solutions with relatively close quality. In order to provide a more comprehensive comparison of their performance, including computation time and solution quality compared to the optimal, more detailed numerical results are shown in Table 4.2.

Table 4.2 includes results by both the Lagrangian and two-level greedy search heuristics for a series of problems with different dimensions and cost settings. Each row in this table shows the results for one of these problems. The first three columns of each row show the problem size parameters, namely N_s , N_a and K . The ODN topology cost given by the greedy heuristic is presented as C_g , together with the gap between this solution and the lower bound provided by the Lagrangian heuristic (not included in the table). This gap is denoted gap_g . The computation time required by this heuristic is displayed as t_g .

The cost produced by the Lagrangian heuristic is given as C_l , together with the gap between this solution and the lower bound LB , which is denoted gap_l ($gap_l = \frac{C_l - LB}{LB}$). The computation time required by this heuristic is presented as t_l , and the costs for each type of resources are denoted c_s , c_d , c_a for server, storage and access

costs, respectively.

Among the problems considered, we were also able to solve a number of small-size ones for exact (optimal) solutions using the mixed integer programming solver provided by the Cplex optimization package. The exact solutions obtained are displayed in the *Cplex* column, together with the computation time t_{cplex} . From the results shown in Table 4.2, the need for heuristic solutions is evident, as solving the provisioning model to optimality is only practical for very small problems. For problems with sizes of 30 nodes, 100 items (i.e. a model of approximately 23000 continuous and 1500 binary variables) and above, no exact solutions were obtainable within a reasonable time, or within about 100 hours' continuous computation, as a rule of thumb used in these experiments. In contrast, heuristic procedures were able to handle problems with orders of magnitude larger number of variables although at a small loss of solution quality.

Each of the two heuristics examined here also appears to have its own strengths and weaknesses. In terms of solution accuracy, the Lagrangian heuristic performs slightly better, with results within 36% from the lower bound in all experiments. Although the gap between the heuristic solution and lower bound varies between experiments, we have not observed any clear trends whether the heuristic solution quality depends on either input parameters or fractional costs. As can be seen from the table, even for problems with very similar cost break-downs, this gap could still vary significantly. However, at small problem sizes, where it is possible to obtain the exact the optimal solution through Cplex, it appears that the heuristic solution is actually very close (within 5% difference) to the optimal solutions. This suggests that although the heuristic does not always provide a good assurance (i.e. bounds) on solution quality, the resulting solution is reasonably close to the optimal.

The two-level greedy search heuristic, on the other hand, gives similar but slightly lower solution quality in most cases, as also illustrated earlier in Figure 4.1. It also has worse worst-case solution quality, which stands at 51% from the lower bound in our experiments compared to the 36% gap in the worst Lagrangian solution.

In terms of computation time, the Lagrangian heuristic performs worse than the greedy heuristic, with computation time exceeding 20 hours as problem size approaches 30 nodes and 1000 items. This is due to the fact that subproblem ($P2$)

Table 4.2 Numerical Results for Web Content ODN Provisioning

Problem size			Two-level greedy heuristic			Lagrangian heuristic						Cplex	
N_s	N_a	K	C_g	$gap_g(\%)$	t_g	C_l	$gap_l(\%)$	$c_s(\%)$	$c_d(\%)$	$c_a(\%)$	t_l	C_{cplex}	t_{cplex}
10	10	10	91781	5	2s	90308	9	16	28	56	27s	86004	12m
10	10	10	78394	9	2s	76223	15	40	14	46	50s	72030	8m
10	10	10	45362	51	2s	31665	12	87	7	6	68s	30010	17m
10	10	10	545970	1	2s	548382	3	5	2	93	10s	542234	15m
15	15	100	568313	17	6m	549558	13	3	68	29	50m	-	-
15	15	100	106594	5	7m	115976	15	2.	56	42	50m	-	-
15	15	100	222781	23	7m	215880	2	66	80	19	48m	-	-
15	15	100	142850	29	7m	150579	36	0.1	99	0.9	56m	-	-
15	15	1000	798656	11	18m	887503	23	3	80	13	13h	-	-
15	15	1000	470674	38	16m	462129	36	42	40	18	26h	-	-
15	15	1000	565316	28	16m	493909	12	12	53	38	21h	-	-
54	54	100	239831	18	20m	221540	9	35	20	45	17h	-	-
54	54	100	397502	21	24m	348221	6	22	19	59	16h	-	-
54	54	100	243349	25	19m	231658	19	15	62	23	22h	-	-
54	54	1000	230500	-	5h	-	-	-	-	-	-	-	-

has a large number of variables in the order of $N_s \times N_a \times K$, thus although it is solvable, performance degrades quickly as problem size increases. The subproblems P_{trans} (4.18) that the greedy heuristic has to solve, on the other hand, are much simpler. As a result, this heuristic has the significantly lower computation time. For example, it can be seen from Table 4.2 computation time t_g is under 2% of computation time t_l for problems of 30 nodes and 1000 items.

From these observations, it seems that the Lagrangian heuristic is the more suitable choice if the problem size is not too large. For very large problems, the greedy heuristic is a better candidate if a feasible solution can be found. However, as explained at the end of Section 4.4.2, if available server resources are very close to the required capacities, the greedy heuristics might not be able to find a feasible solution, even if one exists, in which case the Lagrangian heuristic must be used.

4.7 Conclusion

In this chapter we have addressed the ODN provisioning problem for web content distribution. Based on the ODN provisioning framework developed earlier in Chapter 3, the provisioning problem was formulated as an optimization model that looks for the optimal ODN topology, content replica placement and request routing patterns that deliver a certain required access latency at the minimum total cost. This ODN provisioning model is significantly different from existing models in CDN literature, not only because of the characteristics of the shared infrastructure environment but also because we have aimed to capture the request routing and load balancing feature and avoid the over-simplifications that are seen in many of the existing CDN studies, such as modeling servers with no or only storage-based capacities.

The formulated optimization problem was proven to be an NP-hard problem and as a result heuristic procedures were developed to find near optimal solutions. Three different approaches, including a Lagrangian relaxation heuristic, a two-level greedy search heuristic and a CDN-based greedy search heuristic, were developed, implemented and tested in a number of scenarios with different network topologies and cost factors.

The resulting numerical results have demonstrated that it is extremely important to

develop efficient heuristics for this ODN provisioning problem. Firstly, solving the optimization exactly is only practical for very small problems. The proposed Lagrangian relaxation and two-level greedy search heuristics, on the other hand, could handle problems orders of magnitude larger, although at a small loss of solution quality. Secondly, the performance of the heuristics could have a great impact on the resulting ODN topology cost. For example, using the Lagrangian relaxation or two-level greedy search heuristics, the application provider would be able to find solutions reasonably close to optimal, while using the CDN-based greedy heuristic for the same purpose may lead to topologies up to 2.5 times more costly.

We have also learned that the Lagrangian heuristic and two-level greedy search heuristic have their own strengths and weaknesses. The Lagrangian heuristic produced better topologies, which cost within 36% of optimal, but requires significantly more computation time especially for large problems. The two-level greedy search heuristic is significantly faster, but has slightly worse solution quality, with worst solution being 51% away from optimal. However, unlike the Lagrangian heuristic, this greedy heuristic does not guarantee to find solutions to all feasible provisioning problems.

Chapter 5

Provisioning ODN for Pay-Per-View Content Distribution

5.1 Introduction

This chapter examines the ODN provisioning problem in a pay-per-view content distribution scenario. The basic concept in pay-per-view content distribution is to allow users to freely choose what they want to access and only pay for what they have accessed. This is a business model that originated from and is often found in digital television, where subscribers are able to view extra programs such as movies or sport matches outside their subscription through pay-per-view mechanisms.

In this chapter we focus on the Internet-based distribution of pay-per-view static content that could be replicated in ODN servers, such as recorded TV programs or movies, assuming a business model where the application provider also owns the content and is able to select which movies or recorded shows should be replicated based on their popularity and potential revenues.

While pay-per-view content distribution is a promising application that has already been adopted in the digital television domain, no existing works in current literature has considered its deployment over the Internet in CDN environments. Existing Internet pay-per-view content distribution literature mainly look at the accounting, billing, security or implementation issues of pay-per-view services, with examples in (Cunningham and O'Mahony, 1995; Lee, 2000; Domingo-Ferrer and Martinez-Balleste, 2002; Joo, 2003). Current CDN literature, on the other hand, as reviewed

in Chapter 2, has a small number of studies on multimedia distribution in CDN, for example (Almeida et al., 2002; Yang and Fei, 2003; Cahill and Sreenan, 2003), but none on pay-per-view applications.

Considered within an ODN framework, a pay-per-view content distribution application brings into the provisioning process a number of interesting and unique issues that need to be addressed. The provisioning process would now also need to consider and assess the “worth” of content items based on their demand, resource consumption and potential revenue. As the application provider also owns the content and is able to select whether to replicate each content item, those with low revenue or ratings would probably not get distributed at all. Similarly, if the available distribution resources are limited, the application provider would also need to give priority to items with high revenue and ratings at the expense of other low revenue/rating items. As content items are replicated selectively, the set of replicated items is not known in advance and the provisioning process could no longer aim to minimize the total resource cost. Instead, a profit-based resource provisioning model with selective content replication would be more suitable to pay-per-view content distribution applications.

This chapter is organized as follows. In Section 5.2 the formulation of an ODN provisioning model suitable for pay-per-view content applications will be developed, which aims to maximize the overall profit by selectively replicating available content items. We will also demonstrate the need for this new model by comparing its results with alternative models where content items are not selectively replicated. As the formulated ODN provisioning model is an NP-class problem, which is proven in Section 5.2.5, we will then develop an efficient heuristic procedure that is able to find near optimal solutions within reasonable computation time. The proposed heuristic, which will be presented in Section 5.4, is also based on Lagrangian relaxation and subgradient optimization. Due to the structure of the problem, we have been able to decompose it efficiently into very small subproblems, which gives a better heuristic scalability compared to that in the web ODN model in the previous chapter. Section 5.5 will describe the implementation of the heuristic and discuss the numerical results obtained from this implementation. Finally, major conclusions from the studies in this chapter will be presented in Section 5.6.

5.2 Provisioning Problem Formulation

5.2.1 Assumptions

In formulating this pay-per-view ODN provisioning problem, we assume that content items such as audio or video files are replicated in full. Although a number of studies in Video on Demand literature have advocated partial caching of multimedia content at proxy servers, a study by (Almeida et al., 2002) has shown that in a CDN environment, most of the time full file replication is actually better than partial replication. Besides, we believe that with the current rate of growth in storage capacities and reduction in price, full file replication would be a feasible approach.

Another issue related to modelling the distribution of multimedia content is the fact that content items are typically large and thus each user session would last a certain period of time, unlike in the distribution of web content. In Video on Demand literature, this is usually taken into account by modelling server load using probabilistic functions and complex queueing theory formulae. In this ODN provisioning model, however, in order to make the problem more tractable we assume that content demand could be estimated in terms of average number concurrent sessions required for each content item.

5.2.2 Input Parameters

The problem is modelled in a graph $G = \langle V, E \rangle$, where V is the set of nodes and E is the set of edges. Two subsets of nodes are access nodes V_a and potential server nodes V_s , where $V = V_a \cup V_s$, $N_a = |V_a|$ and $N_s = |V_s|$. An access node is where end users reside, e.g. an ISP access network. A potential server node represents a location that resources can be leased and a virtual server can be established, e.g. a server farm owned by an infrastructure provider.

The costs associated with leasing a virtual server at location $i \in V_s$ and delivering content from this server would include:

- c_i : server cost, i.e. the cost involved in establishing and maintaining the virtual server.

- d_i : unit storage cost. The total storage cost would be proportional to the amount of content replicated at the server.
- b_i : unit delivery bandwidth cost.
- p_i : unit processing power cost.
- B_i : maximum server delivery bandwidth.

Note that we have used delivery bandwidth limits to model server capacities, as this is likely the most critical resource bottleneck in the delivery of multimedia content. However, server capacities could also be modelled based on server processing or number of sessions without significant changes to the mathematical formulation.

Each edge $e \in E$, represents a network link and has a weight that reflects the latency between the two nodes. The latency τ_{mi} between any node $i \in V_s$ and $m \in V_a$ is thus known and pre-calculated using shortest path algorithms on graph G .

Content to be considered for replication and delivery are represented by a population of K items, with item k having the following properties:

- α_k : storage requirement
- γ_k : delivery bandwidth consumption
- β_k : processing power consumption
- T_k : latency requirement
- ρ_k : potential revenue
- λ_{mk} average demand from customer node m

The demand λ_{mk} of a content item reflects its popularity at each customer access node. As mentioned in Section 5.2.1, this is measured by the estimated number of users from node m concurrently accessing content item k . As a result, the delivery resource requirements as well as potential revenue (γ_k , β_k and ρ_k) are all measured per unit time instead of over the whole duration of a given content item. Even though at a particular instant different users may actually be accessing different parts of a

certain content, for example different positions within a movie clip, it is reasonable to assume that the same bandwidth and processing requirement applies.

5.2.3 Decision Variables

To formulate this provisioning problem the following binary and continuous decision variables are defined:

$$y_i = \begin{cases} 1 & \text{if site } i \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

$$u_k = \begin{cases} 1 & \text{item } k \text{ is replicated} \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

$$x_{ik} = \begin{cases} 1 & \text{if item } k \text{ is replicated at site } i \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

$$r_{mi}^k \in [0, \lambda_{mk}] \quad (5.4)$$

The continuous variable r_{mi}^k is used to indicate the fraction of accesses for item k from customer m that should be directed to site i . Note that as full file replication is assumed, the set of binary variables x_{ik} is adequate to fully determine how content items are replicated in the whole ODN. Furthermore, binary variables (u_k) have been defined in order to include profit-based content selection into the provisioning process.

A complete list of variables and parameters defined in this formulation is reproduced in Table 5.1 for convenient reference.

5.2.4 Formulation

The provisioning problem is formulated as an optimization whose objective is to determine which items should be distributed through the ODN and find an optimal distribution network topology. This includes determining server locations and capacities, item replication and user redirection patterns so that total service profit (i.e. total revenue less total cost) is maximized, while available resources are not exceeded and delivery latency requirements are satisfied. Based on the variables and parameters, the problem is formulated as the following mixed integer linear programming

model:

$$\begin{aligned}
 (Q) \quad \text{Max} \quad & \underbrace{\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \rho_k}_{\text{revenue}} - \underbrace{\sum_{i \in V_s} \sum_{k \in K} x_{ik} d_i \alpha_k}_{\text{storage cost}} - \\
 & \underbrace{\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (p_i \beta_k + b_i \gamma_k)}_{\text{bandwidth and cpu cost}} - \underbrace{\sum_{i \in V_s} y_i c_i}_{\text{site cost}}
 \end{aligned} \tag{5.5}$$

subject to:

$$\sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \gamma_k \leq y_i B_i, \forall i \tag{5.6a}$$

$$\sum_{i \in V_s} r_{mi}^k = u_k \lambda_{mk}, \forall m, k \tag{5.6b}$$

$$r_{mi}^k (\tau_{mi} - T_k) \leq 0, \forall m, i, k \tag{5.6c}$$

$$r_{mi}^k \leq \lambda_{mk} x_{ik}, \forall m, i, k \tag{5.6d}$$

$$x_{ik} \leq y_i, \forall i, k \tag{5.6e}$$

$$x_{ik}, u_k, y_i \in \{0, 1\} \tag{5.6f}$$

$$r_{mi}^k \in [0, \lambda_{mk}] \tag{5.6g}$$

In this model, constraint (5.6a) enforces the server capacity at each potential site, (5.6b) ensures all requests for replicated content are allocated to virtual servers. Constraint (5.6c) means that item k can only be delivered from server i to user m if the latency between them (τ_{mi}) is not more than its minimum requirement T_k . Constraints (5.6d) and (5.6e) set the dependencies among variables r, x and y , namely an item can only be delivered from a location where it is replicated and it can only be replicated where a server has been established. The relationship $u_k \geq x_{ik}$ (i.e. replications are only made if the item is selected) is implied by a combination of constraints (5.6b) and (5.6d).

5.2.5 Problem Complexity

The provisioning model above has the following numbers of variables and constraints:

Table 5.1 Parameters and Variables for Pay-Per-View ODN Provisioning Model

Notation	Meaning
<i>Resource parameters</i>	
b_i	unit delivery bandwidth cost at server i
c_i	server start cost
d_i	server unit storage cost
p_i	server unit processing cost
B_i	server delivery capacity
τ_{im}	distance between server i and access node m
N_s	number of sever nodes, $N_s = V_s $
N_a	number of access nodes, $N_a = V_a $
i	subscript for server nodes
m	subscript for access nodes
<i>Content parameters</i>	
α_k	size (storage requirement) of content item k
γ_k	delivery bandwidth consumption
β_k	delivery processing power consumption
λ_{mk}	request rate for item k from access node m
ρ_k	service revenue per access
T_k	latency requirement
k	subscript for content items
<i>Decision variables</i>	
y_i	server establishment indicator, $y_i \in 0, 1$
x_{ik}	content replication indicator, $x_{ik} \in 0, 1$
u_k	content selection indicator, $u_k \in 0, 1$
r_{mi}^k	request redirection ratio, $r_{mi}^k \in [0, \lambda_{mk}]$

- Binary variables: $N_s + K + N_s K$, i.e. $O(N_s K)$
Continuous variables: $N_a N_s K$, i.e. $O(N_a N_s K)$
Total number of variables: $N_s + K + N_s K + N_a N_s K$, i.e. $O(N_a N_s K)$
- Constraints: $2N_a N_s K + N_a K + N_s K + N_s$, i.e. $O(N_a N_s K)$

This model is also an NP class problem, as in some special cases it can be reduced to the binary knapsack problem (Martello and Toth, 1990), which has been well known to be NP-complete, as seen in the following:

Consider a special case of this model with only one server site, zero resource costs and no performance constraints (i.e. $N_s = 1$, $b_i = 0$, $c_i = 0$, $d_i = 0$, $p_i = 0$,

and $L_k = \infty$). With subscript i removed for convenience and redundant constraints eliminated, the problem can be written as:

$$\text{Max} \sum_{m \in V_a} \sum_{k \in K} \rho_k r_m^k$$

subject to:

$$\sum_{m \in V_a} \sum_{k \in K} \gamma_k r_m^k \leq B$$

$$r_m^k = \lambda_{jk} u_k, \forall j, k$$

$$r_m^k \in [0, \lambda_{mk}] \forall m, k$$

$$u_k \in \{0, 1\}, \forall k$$

By replacing r_m^k with $\lambda_{mk} u_k$, the model can then be transformed on to:

$$\text{Max} \sum_{k \in K} u_k (\rho_k \sum_{m \in V_a} \lambda_m^k)$$

subject to:

$$\sum_{k \in K} u_k (\gamma_k \sum_{m \in V_a} \lambda_m^k) \leq B$$

$$u_k \in \{0, 1\}, \forall k$$

which is a classical Knapsack problem.

5.3 Alternative Provisioning Models

In this section we aim to demonstrate the need for a profit-based ODN provisioning model with integrated content selection by evaluating the impact on the profit of the pay-per-view application if other ODN provisioning models were used. For comparison purpose, we consider the following two alternative approaches:

1. *No content item selection:* In this approach, the provisioning process would simply assume that all content items are to be replicated for delivery and aims

to find the ODN configuration with the minimum cost to support this. The formulation of such a provisioning model is as follows:

$$(Q_{a1}) \quad \text{Min} \quad \underbrace{\sum_{i \in V_s} \sum_{k \in K} x_{ik} d_i \alpha_k}_{\text{storage cost}} + \underbrace{\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (p_i \beta_k + b_i \gamma_k)}_{\text{bandwidth and cpu cost}} + \underbrace{\sum_{i=1}^N y_i c_i}_{\text{site cost}} \quad (5.7)$$

subject to:

$$\begin{aligned} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \gamma_k &\leq y_i B_i, \forall i \\ \sum_{i \in V_s} r_{mi}^k &= \lambda_{mk}, \forall m, k \\ r_{mi}^k (\tau_{mi} - T_k) &\leq 0, \forall m, i, k \\ r_{mi}^k &\leq \lambda_{mk} x_{ik}, \forall m, i, k \\ x_{ik} &\leq y_i, \forall i, k \\ x_{ik}, y_i &\in \{0, 1\} \\ r_{mi}^k &\in [0, \lambda_{mk}] \end{aligned}$$

2. *Stand-alone content item selection:* In this approach, instead of performing content selection within the provisioning model as in model (Q) or no content selection at all as in model (Q_{a1}), we carry out a content item selection process independent of the provisioning process.

In this approach, each item is assessed based on its estimated potential revenue and resource cost. Given the input information described in Section 5.2.2, these parameters for content item k can be calculated as follows:

$$\begin{aligned} \text{revenue}(k) &= \sum_{m \in V_a} \lambda_{mk} \rho_k \\ \text{cost}(k) &= \sum_{m \in V_a} \lambda_{mk} (\bar{p} \beta_k + \bar{b} \gamma_k) + \bar{d} \chi_k \end{aligned}$$

where \bar{p} , \bar{b} , and \bar{d} are average processing power, bandwidth and storage cost over all server sites, respectively, and χ_k is the minimum number of replicas

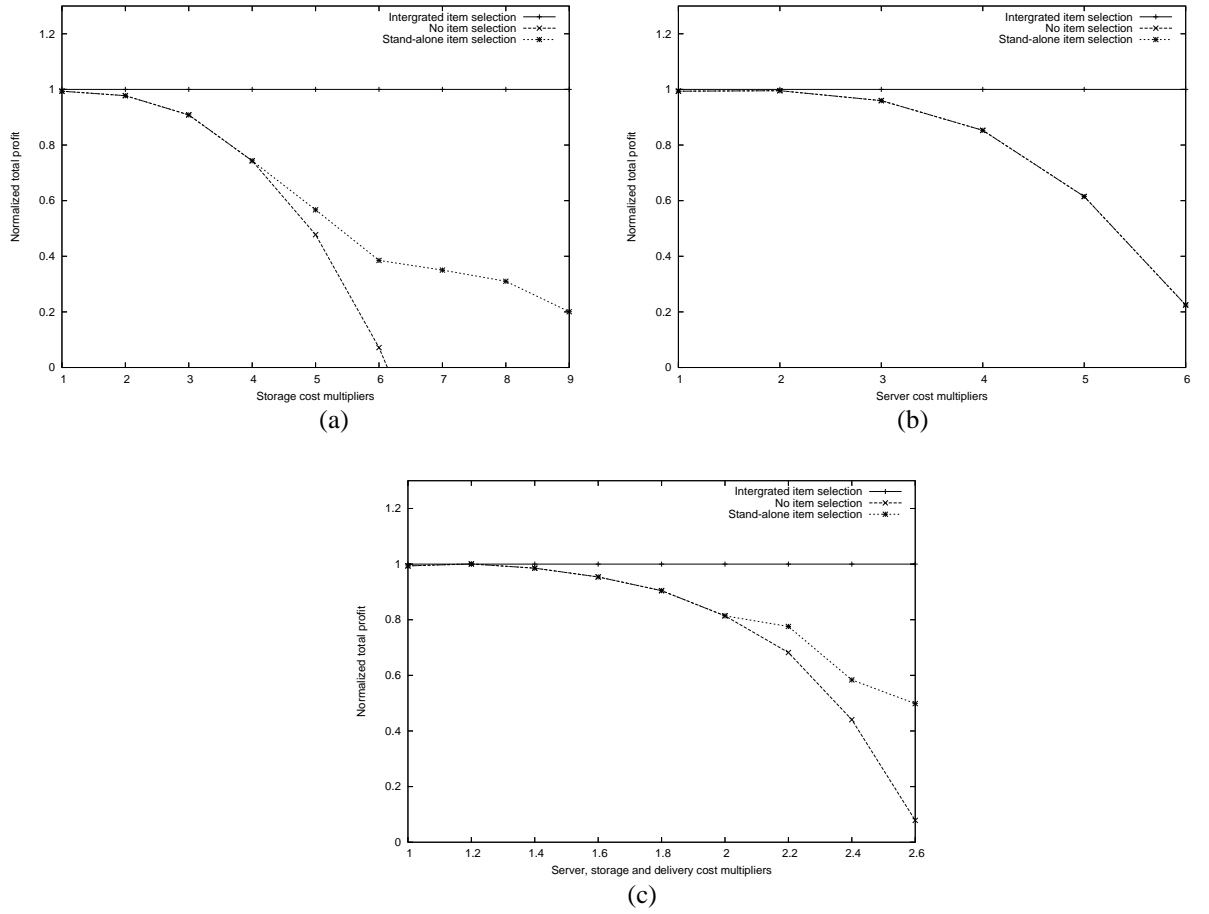


Figure 5.1 Impact of ODN provisioning model selection on pay-per-view content distribution profit

of item k that have to be deployed so that each access nodes has one replica of the item within distance T_k . A simple optimization problem that could be solved for χ_k is described in Appendix C.

Each content item k will then be selected for replication if $revenue(k) > cost(k)$, and not selected otherwise. The set of items to be replicated $K^* = \{k \in K | revenue(k) > cost(k)\}$ can then be used as an input to the cost minimization problem model (Q_{a1}) above in order to find a suitable ODN topology.

In order to compare the performance of these alternative provisioning models, we analyzed their optimization output for a series of problems with different revenue and cost settings. The results are shown in Figure 5.1, where the total profit result-

ing from the ODN topology created by each alternative ODN provisioning model is normalized by the profit obtained with the provisioning model (Q), in order to plot results from different problems on the same graph.

In the experiment shown in Figure 5.1, we start from a problem with high content revenue and low resource cost, so that all items are selected and replicated in all three provisioning models. We then gradually increase resource costs compared to revenue in order to make a number of items not worth replicating. This is done by multiplying storage cost (Figure 5.1.a), or server cost (Figure 5.1.b) or all costs (Figure 5.1.c) with gradually increasing multipliers. This is because we predict that the differences between the different approaches would become more apparent as more content items become expensive and should not be replicated, which makes having a suitable content selection scheme more critical.

As expected, the graphs in Figure 5.1 show widening differences between the profits produced by provisioning model (Q), which performs content item selection as part of the optimization process, and the results by the other two models as resource costs increase. The ODN provisioning model (Q) always perform the best, with the resulting profit many times higher than the other approaches and the provisioning model without content item selection (Q_{a1}) performs the worst. The provisioning scheme with stand-alone item selection does have better performance compared to no item selection, especially when storage cost is high. However, its performance could still be up to 5 times worse than that of the profit-based model.

In short, the results in this section have demonstrated that the profit-based ODN provisioning model with integrated content item selection produces significantly better performance than the alternative provisioning models considered, and thus is the most suitable provisioning model for pay-per-view applications in ODN environment. Having determined that, in the next section we will develop a heuristic procedure that allows us to solve this pay-per-view ODN provisioning problem, which has been proven to be NP-complete, efficiently.

5.4 Heuristic Procedure

Since the ODN provisioning model, as developed in Section 5.2, is NP-complete, it cannot be solved efficiently for realistically large networks. Thus in this section we will develop a heuristic based on Lagrangian relaxation in order to find its near optimal solutions within reasonable time.

5.4.1 Applying Lagrangian Relaxation

The original model in (Q) in 5.5 is first transformed into the following equivalent form, which minimizes the negative of the profit:

$$(Q_{eq}) \quad \text{Min} \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (b_i \gamma_k + p_i \beta_k - \rho_k) + \sum_{i \in V_s} \sum_{k \in K} x_{ik} d_i \alpha_k + \sum_{i \in V_s} y_i c_i \quad (5.8)$$

subject to: (5.6a) to (5.6g) Solving this equivalent problem will give the solution to original problem (Q) and the negative of its objective function. This new equivalent problem will now be solved with a heuristic based on Lagrangian relaxation and sub-gradient optimization.

Using Lagrangian relaxation, constraints (5.6a) and (5.6b) are eliminated and following terms are added to Q_{eq} 's objective function:

$$\sum_{i \in V_s} w_i \left(\sum_{m \in V_a} \sum_{k \in K} \gamma_k r_{mi}^k - y_i B_i \right) \quad \text{and} \\ \sum_{m \in V_a} \sum_{k \in K} \mu_{mk} \left(\sum_{i \in V_s} r_{mi}^k - u_k \lambda_{mk} \right)$$

where $w_i, \mu_{mk} \in \mathbb{R} | w_i \geq 0$ are Lagrangian multipliers.

Constraints (5.6a) and (5.6b) have been chosen to be relaxed as their elimination would allow a high degree of decomposition, where the model can be broken into extremely simple and small subproblems, yet reconstructing a feasible solution from solving these problems still remains fairly simple.

After re-arranging the objective function, the relaxed model has the following form:

$$(Q_{LR}) \quad \text{Min} \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \underbrace{(b_i \gamma_k + p_i \beta_k - \rho_k + w_i \gamma_k + \mu_{mk})}_{A_{mi}^k} + \\ \sum_{i \in V_s} \sum_{k \in K} x_{ik} \underbrace{d_i \alpha_k}_{H_{ik}} + \sum_{i \in V_s} y_i \underbrace{(c_i - w_i B_i)}_{D_i} - \sum_{k \in K} u_k \left(\sum_{m \in V_a} \mu_{mk} \lambda_{mk} \right) \quad (5.9)$$

subject to:

$$\begin{aligned}
r_{mi}^k(\tau_{mi} - T_k) &\leq 0, \forall m, i, k \\
r_{mi}^k &\leq \lambda_{mk} x_{ik}, \forall m, i, k \\
x_{ik} &\leq y_i, \forall i, k \\
x_{ik}, u_k, y_i &\in \{0, 1\} \\
r_{mi}^k &\in [0, \lambda_{mk}]
\end{aligned}$$

This can then be broken into the following independent subproblems:

$$(Q1) \text{ Max } \sum_{k \in K} u_k \left(\sum_{m \in V_a} \mu_{mk} \lambda_{mk} \right) \quad (5.10)$$

subject to:

$$u_k \in \{0, 1\}$$

and:

$$(Q2) \text{ Min } \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k A_{mi}^k + \sum_{i \in V_s} \sum_{k \in K} x_{ik} H_{ik} + \sum_{i \in V_s} y_i D_i \quad (5.11)$$

subject to:

$$\begin{aligned}
r_{mi}^k(\tau_{mi} - T_k) &\leq 0, \forall m, i, k \\
r_{mi}^k &\leq \lambda_{mk} x_{ik}, \forall m, i, k \\
x_{ik} &\leq y_i, \forall i, k \\
x_{ik}, y_i &\in \{0, 1\}; r_{mi}^k \in [0, \lambda_{mk}]
\end{aligned}$$

with A_{mi}^k , H_{ik} , and D_i as defined in (5.9).

Both subproblems (Q1) and (Q2) can be solved easily through simple inspection, as described in detail in Appendix B.1. For each value of multipliers w_i and μ_{mk} , solving these subproblems gives a solution to the corresponding relaxed problem $Q_{LR}(\mu, w)$. The resulting objective function value, $Obj(Q_{LR}(\mu, w))$, is a lower bound of the optimal objective of Q_{eq} , due to the properties of Lagrangian relaxation (Glover and Laguna, 1993):

$$Obj^*(Q_{LR}(\mu, w)) \leq Obj^*(Q_{eq}) \quad (5.12)$$

where $Obj^*(.)$ denotes the optimal objective function value of an optimization problem.

5.4.2 Constructing a Feasible Solution

A solution to the relaxed problem Q_{LR} may not be a feasible solution to Q_{eq} , since constraints (5.6a) and (5.6b) may have been violated. However, this relaxed solution is used as a starting point to reconstruct a feasible solution to Q_{eq} . The strategy used here is to accept the service replication parameters (i.e. variables x_{ik} , y_i , u_k) and adjust the user redirections (i.e. variables r_{mi}^k) so that the relaxed constraints are again satisfied. Thus the following solution recovery procedure is used:

1. Initialize:

- rank items in decreasing order of priority.
- initialize server capacities: $B_i^{residual} = B_i, \forall i$
- start with first item: set $k = 1$

2. For each item k , find the request routing pattern (represented by values of variables r_{mi}^k), that results in minimum total delivery cost for that item. This is done assuming the item is already replicated at locations specified by variables x_{ik} . The optimal request redirection variables can be found by solving the following problem, where subscript k in variables has been dropped for clarity:

$$\text{Min} \sum_{i \in V_s} \sum_{m \in V_a} r_{mi} (b_i \gamma + p_i \beta) \quad (5.13)$$

subject to:

$$r_{mi} \gamma \leq y_i B_i^{residual}, \forall i \quad (5.14)$$

$$\sum_{i \in V_s} r_{mi} = u \lambda_m, \forall m \quad (5.15)$$

$$r_{mi} (\tau_{mi} - T) \leq 0, \forall m, i \quad (5.16)$$

$$r_{mi} \leq \lambda_m x_i, \forall m, i \quad (5.17)$$

$$r_{mi} = 0, \text{ if } r_{mi}^{relaxed} = 0 \quad (5.18)$$

$$r_{mi} \in [0, \lambda_m] \quad (5.19)$$

Note that in this problem, constraint (5.18) forces a variable r_{mi} to be zero if the corresponding variable is zero in the relaxed solution. In other words, request routing paths are searched from the set used by the relaxed solution only.

If the relaxed solution rules out delivering item k from server i to customer m , it is then not considered in the solution recovering process. This constraint has been introduced through experiments, as numerical results showed that the reconstruction process yields slightly but consistently better feasible solutions with its inclusion.

With x , y , u already defined, the above problem is a pure linear programming problem and can be solved by a standard technique such as simplex (Nemhauser and Wolsey, 1999), which can be done by optimization package Cplex (Cplex, 2001). However, the problem's special structure also allows it to be transformed to a special case of a max flow min cost problem (Ahuja et al., 1993), which has been exploited in this work to solve this problem even more efficiently with a special adaptation of the path augmenting algorithm (Ahuja et al., 1993).

3. Update residual server capacity if the above problem is feasible (i.e. item k is replicated)

$$B_i^{residual} = B_i^{residual} - \sum_{m \in V_a} r_{mi} \gamma_k$$

4. Proceed to next item:

- $k = k + 1$
- go back to Step 2

For ordering items in Step 1, different item ranking criteria have been experimented with, including:

1. revenue per request: $priority(k) = \rho_k$
2. estimated profit per request: $priority(k) = \rho_k - dcost_k$
 where $dcost_k$ is the estimated delivery cost for this item, $dcost_k = \bar{b} \cdot \gamma_k + \bar{p} \cdot \beta_k$,
 \bar{b} and \bar{p} are mean unit costs for delivery bandwidth and processing, respectively.
3. potential net revenue: $priority(k) = \sum_m \lambda_{mk} * \rho_k$

4. potential net profit: $priority(k) = \sum_m \lambda_{mk}(\rho_k - dcost_k)$

5. revenue/delivery cost ratio: $priority(k) = \frac{\rho_k}{dcost_k}$

6. profit/computation cost ratio: $priority(k) = \frac{\rho_k - dcost_k}{\bar{b}}$

the last criteria (profit/computation ratio) was experimented with because the optimization model can be seen to have the embedded structure of a knapsack problem, where the gain is profit and the weight is processing resource, which is limited by server capacities.

The solution (x, y, u, r) produced by this procedure will then be a feasible solution to problem (Q_{eq}) . Obviously, the corresponding objective function value $Obj(Q_{eq})$ cannot be better than the optimal objective:

$$Obj(Q_{eq}(x, y, u, r)) \geq Obj^*(Q_{eq}) \quad (5.20)$$

5.4.3 Subgradient Procedure

From properties (5.12) and (5.20), it can be seen that for any value of the Lagrangian multipliers, by solving the relaxed problem a lower bound to the optimal value of (Q_{eq}) is obtained. Using this result to reconstruct a feasible solution, an upper bound on that optimal value is found. The difference between the upper and lower bounds then gives an indication on how far from optimal the reconstructed feasible solution is.

Therefore, instead of solving the full problem (Q_{eq}) , the Lagrangian heuristic solves a series of relaxed problems for different multiplier values iteratively. In each iteration, a lower bound and a feasible solution are obtained. The multipliers are adjusted to search for better and better solutions, i.e. reduce gap between the bounds. This is again achieved using the well-known subgradient optimization technique (Glover and Laguna, 1993). Detailed description of the procedure, as applied in this heuristic, is included in Appendix A.3.

Due to the equivalence of problems (Q) and (Q_{eq}) , the best feasible solution found is a heuristic solution to original problem Q and the magnitude of the best lower bound gives an upper bound on (Q) 's objective.

5.5 Heuristic Results

The proposed Lagrangian heuristic has been implemented in C++ and tested on network topologies generated by the GT-ITM topology generator (Calvert et al., 1997). In each topology, server capacities, content requests and resource costs are randomly generated and different cost combinations in which resource costs are varied across a wide range are experimented. Note that although the heuristic and its implementation solves the equivalent model (Q_{eq}), the results will now be presented and discussed in terms of the original profit-maximization provisioning problem, (Q).

Table 5.2 shows the results obtained from a number of problems with different sizes. Some very small problems were used so that an exact (optimal) solution could be obtained with the mixed integer programming solver provided by optimization package Cplex (Cplex, 2001), which enabled a comparison between heuristic and optimal solution performance.

In the table, each row shows the result from a different problem. The first three columns show problem size parameters, namely N_a , N_s and K . Heuristic results are given in terms of the objective function (i.e. service profit) C_Q and the upper bound UB of this objective. The *gap* between these values, which is calculated by $gap = \frac{UB-LB}{UB}$ indicates the worst case distance from heuristic and optimal results. Fractional costs, as contributed by each type of resources – denoted c_c , c_d , c_p and c_b for server, storage and processing and bandwidth costs, respectively – are also displayed. Exact solutions, if obtainable, are placed in the *Optimal* column. Column gap^* then shows the actual gap between this optimal result and the heuristic results. All computation in these experiments were done in a machine with a 1.4GHz CPU and 512MB of memory. Experimental problems that exceeds 100h's continuous computation time without finishing are considered unsolvable, which are indicated by the “-” symbols in the result table.

5.5.1 Observations

In terms of solution accuracy, these numerical results show that the heuristic has performed well, with heuristic solutions being at most 20% from the upper bound. In contrast with the results seen in the previous chapter, a consistent heuristic per-

Table 5.2 Numerical Results for Pay-Per-View Content ODN Provisioning

Problem size			Heuristic result								Cplex result		
N_s	N_a	K	C_Q	UB	$c_c(\%)$	$c_d(\%)$	$c_p(\%)$	$c_b(\%)$	gap (%)	time	Optimal	gap* (%)	time
10	10	10	19423	21824	14	2	6	78	11	1s	19539	0.6	2s
			46497	48944	70	10	7	13	5	2s	48798	4.7	14s
			36411	38735	26	20	26	28	6	2s	37185	2	73s
10	10	20	4199	4564	88	5	2	5	8	3s	4343	3.3	419s
			8219	10274	24	21	44	11	20	2s	8406	2.2	800s
			14393	16203	2	15	23	60	11.2	5s	-	-	-
54	54	100	56464	57371	8	5	41	46	1.6	8m	-	-	-
			45519	55127	80	3	3	14	17.4	7m	-	-	-
			58877	59526	10	50	10	30	1	7m	-	-	-
60	80	1000	11709	14279	22	74	2	2	18	2.0h	-	-	-
			20759	23065	20	40	20	20	10	1.5h	-	-	-
			19252	22919	6	14	14	66	16	1.0h	-	-	-

formance has been observed with different combinations of component cost breakdowns.

The heuristic solution is seen to be consistently close to the optimal solution (under 5%), as seen in the gap^* column for the first few small problems. These small problem sizes were used so that exact optimal solutions could be obtained with Cplex for comparison purposes.

Experimental results also suggest that how items are ordered during solution recovery process (Section 5.4.2) is important. This has been found by comparing solutions obtained with the different item ordering criteria as mentioned in Section 5.4.2 in a series of problems generated on the same topology, but with increasingly lower server capacities. The intuition behind this comparison is that the only factor affected by different item orders is residual server capacities after the placement of each item. Therefore, when server resources are abundant, item ordering is clearly not significant, but as server capacities shrink, proper content prioritization would become more critical. The experiment examined a series of seven such problems originated from the same topology, and the results obtained from optimizing these problems are presented in Figure 5.2. Interestingly, as can be seen from the figure, the different content ordering methods give very similar results. This suggests that the relaxed solution has already been reasonably effective in selecting items and indicating good request routing paths, which is then taken into account in the solution recovery process.

As a result, all the data presented in this section were obtained with the “total revenue” item ordering method, which has been observed to perform reasonably well in all experiments.

In terms of heuristic scalability (i.e. computation time), the experiments show very encouraging results. While solving the problem to optimality is only practical for small problems of the order of 20 nodes and 20 content items, the Lagrangian heuristic can handle problems of many orders of magnitude larger in size. As an example, for a problem of 120 nodes and 1000 items (i.e. about 61,000 binary variables and 4,800,000 continuous variables), a near optimal solution can be found in the order of hours.

This heuristic is also significantly more scalable compared to the heuristics devel-

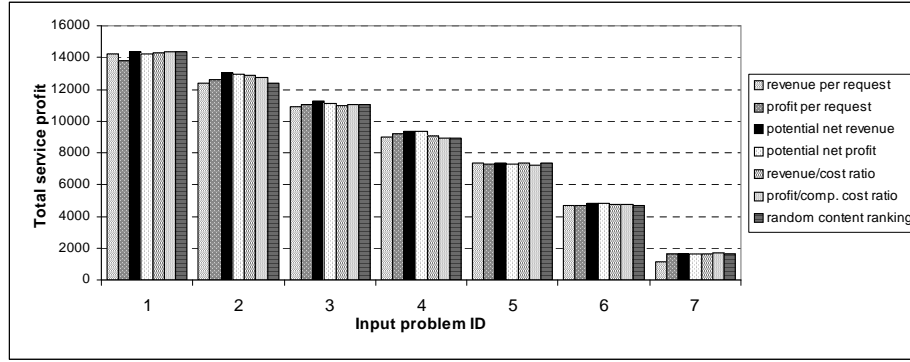


Figure 5.2 Comparing different content ranking methods for problems with progressively decreasing server capacities

oped for provisioning model (P) (4.4) in Chapter 4. This is because the structure of the current problem allows it to be decomposed into extremely small subproblems, which only require simple inspections to solve, as discussed in Section 5.4.1.

Close examination of the heuristic trace also reveals that the heuristic is reasonably quick in approaching its final solution. This can be seen from Figure 5.3, which plots the upper bounds and objective functions (i.e. lower bounds) of some of the sample problems during the first 1000 heuristic iterations. The top curves shows the upper bounds, and the bottom ones are objective functions. Both of these values have been normalized by the corresponding final objective function, so that they can be plotted on the same graph. Bounds from the same problem are plotted with the same graph style. It can be seen from this figure that the most significant solution improvements are made during the first few hundred to a thousand iterations. The heuristic often approaches quickly close to its final solution, and spend the rest of the time making small improvements or simply improving the upper bound (i.e. proving that the current candidate solution is close to optimal). This type of behaviour has been observed with all experiments. As a result, the heuristic has been usually run for a maximum of only 4000 iterations in all experiments.

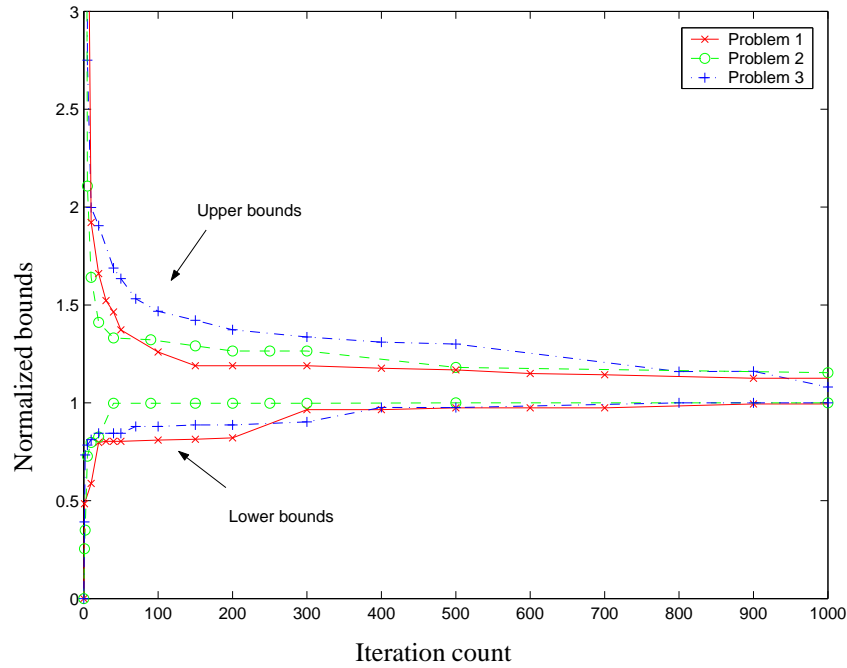


Figure 5.3 Solution bounds for the first 1000 iterations

5.6 Conclusion

This chapter has considered the problem of provisioning ODNs for Internet-based pay-per-view content distribution applications. For this type of application, service revenue comes from the end users and is directly related to the users' demand. Assuming that the application provider also owns the content and is able to select whether or not to distribute each content item, pay-per-view applications introduce new and interesting requirements into the ODN provisioning process. In this scenario content items are not all necessarily replicated but rather selected for distribution based on their profit potential. Similarly, if distribution resources are limited, the application provider would also want to prioritize content items with higher demand/revenue for replication and distribution at the expense of other less profitable ones.

In this chapter the impacts of such distribution policies on the ODN provisioning process has been examined and a suitable provisioning model has been formulated as a mixed integer linear optimization problem that incorporates content item selec-

tion and aims to maximize the overall service profit. By comparing the output of this model and other alternative provisioning approaches that do not incorporate content selection, we have been able to demonstrate that the formulated profit maximization model is the most suitable for pay-per-view ODN provisioning.

As the ODN provisioning model formulated above is an NP-complete problem, we have developed an efficient Lagrangian heuristic to find near optimal solutions. This heuristic exploits the problem's structure to break it down into a series of very small and simple subproblems. Experimental results showed that the heuristic is able to find reasonably good near optimal solutions, which are within 20% from optimal in all experiments, with good computation scalability. For example solutions close to optimal solutions for problems of up to 120 nodes and 1000 content items could be found within by the heuristic in the order of hours. In contrast, looking for the exact solutions was practical only for very small problems of up to 20 nodes.

Given these promising results, there are still a number of outstanding issues that are not addressed in this chapter. Firstly, provisioning scalability may need to be further improved, as even with the current level of computation scalability, the provisioning problem may still have difficulty in dealing with realistically larger problems, for example ODNs with 10,000s or even 100,000s of TV programs or movies to distribute. We will attempt to address this in Chapter 7 by grouping similar content items into clusters which are then treated as a single item, which would drastically reduce the number of items to consider and, as a result, the optimization problem size. Secondly, the provisioning model in this chapter has assumed a rather static estimation of content demand distribution. As the actual demand bounds to change during the application's life time, the provisioned ODN may need to be adjusted accordingly. Towards this end we envisage two possible approaches. The ODN may be adjusted on-the-fly by periodically repeating the above provisioning process using new demand estimates, or by employing a specially designed light-weight optimization that only fine-tunes the current topology and content replication. The dynamic content replacement algorithm for multimedia content in CDN proposed recently in (Cahill and Sreenan, 2003) and (Cahill and Sreenan, 2005) serves a very close purpose. Thus the proposed algorithm, if extended to take into account the characteristics of pay-per-view content, would fit well with our ODN provisioning model.

Chapter 6

Provisioning ODN for Live Multimedia Content Distribution

6.1 Introduction

This chapter looks at the ODN provisioning problem for another class of content applications that we believe have great potential of becoming one of the key players in the future Internet content distribution landscape - the distribution of *live* streaming multimedia such as television or radio content.

Often referred to as the convergence of the Internet and the television, Internet-based distribution of live multimedia content has long been proposed and predicted in both research and industries to be the next killer application (Sandbank, 2001), (Shim and Lee, 2002). The combination is attractive due to the ability to have vast amounts of content choices and the possibility of introducing rich functionalities such as content adaptation, personalization, localization or interactivity.

Early forms of this convergence can be observed in the production of televisions capable of browsing the Internet and PCs that can receive television signals, or the emergence of thousands of live radio and television services over today's Internet (WebTVlist, 2005). However, the ultimate convergence where high quality multimedia is streamed live to the living room through the Internet is still some distance into the future. However, with the current rate of growth of the Internet, the increasing penetration of broadband access and advances in digital multimedia technologies, we are convinced this is coming and are interested in contemplating the

entailed challenges.

In this future, with the introduction of content enhancement features such as adaptation or personalization, distributing high quality live content would place extremely heavy burdens on both network and server resources. As the application gains popularity, a CDN-like globally distributed server and network infrastructure would certainly be required to achieve service scalability. This makes the ODN very suitable as a cost effective and flexible distribution platform.

Using ODNs, the virtual servers would function not only as caching and streaming proxies that deliver content to end users, but also as processing nodes where content enhancement functions, for example personalization, mixing or transcoding, are performed. Inter-server virtual links, on the other hand, would form a backbone over which content is streamed to servers from content sources such as radio or TV stations. Packet duplicating and forwarding functions could also be implemented at servers sites, which effectively provides a consistent overlay multicast capability within the ODN backbone in order to minimize the distribution bandwidth cost. We believe this is desirable because while multicast has been proven to be key factor in achieving bandwidth efficiency in multimedia delivery, it is not expected to be available consistently across the Internet in the foreseeable future.

Compared to content applications previously considered, the distribution of live multimedia has significantly different requirements that warrant the development of a different ODN provisioning model. Firstly, the possibility of using each server location not only depends on server resources, such as computation, storage and Internet bandwidth, but also whether a distribution path with adequate quality could be established from the content source. Secondly, live content is continuously streamed instead of replicated statically at servers, thus bandwidth usage in the ODN backbone links would contribute to a significant portion of the overall cost, especially if QoS-assured links are required, and thus cannot be ignored in the cost function. As a result, the creation of backbone virtual links and routing of multicast distribution trees must be addressed alongside server placement and request routing.

In this chapter, a provisioning model that addresses these issues will be formulated as a mixed integer programming problem, which is also an NP-hard problem. An efficient Lagrangian heuristic will therefore be developed to find near optimal solu-

tions instead of solving the problem for exact solutions.

By periodically repeating this resource provisioning process, the ODN virtual topology could be adjusted to suit changes in service demand. However, the rate of such changes, e.g. once in the order of hours, days or weeks, depends on the flexibility of the underlying infrastructure. In order to respond to demand fluctuations in the meantime, we will further formulate an online optimization problem that adjusts the distribution process, including redirecting clients or rerouting distribution paths, in order to better utilize the currently allocated resources. The performance of this online optimization is also expected to give an insight into the impact of demand estimation inaccuracies on service provisioning solution quality and subsequent service performance. Different online optimization models will be developed and examined, including a centralized global optimization that re-optimizes the whole distribution process, and a local optimization that is carried out independently at each server.

We believe our study is covering an interesting area that has not previously been addressed. Although there have been a vast amount of existing research on Internet distribution of both archived and live streaming content, for example those in (Chan and Tobagi, 2001; Allen, 2001; Sen et al., 2002; Tran et al., 2003; Hama et al., 2004) and references therein, most of these studies concentrated on single server scenarios. Research into the distribution of live streaming multimedia in a distributed-server, CDN-like environment, has been lacking.

The chapter will be organized as follows. Section 6.2 will present the mathematical formulation of the provisioning problem, whose complexity will be examined in Section 6.2.4. An efficient heuristic based on Lagrangian relaxation will then be developed in Section 6.3. Section 6.4 will describe the online optimization process and present the formulation of a number of alternative online optimization schemes, including full optimization, partial optimization and local optimization. The numerical results obtained from these models are then presented and analyzed in Section 6.5. Finally contribution summary and concluding remarks are included in the last section of the chapter, Section 6.6.

6.2 Provisioning Problem Formulation

Given a multitude of potential server and transport services that can be leased from different server and network providers, this provisioning process will determine where and how much server and network resources are leased, so as to satisfy the service's demand and at the same time minimize the operational cost.

6.2.1 Input Parameters

The network is modelled as a graph $G = \langle V, E \rangle$, where V is the set of nodes and E is the set of directed edges. Three subsets of nodes are defined - access nodes V_a , potential server nodes V_s and content source nodes V_c , where $V = V_a \cup V_s \cup V_c$, $N_a = |V_a|$, $N_s = |V_s|$ and $N_c = |V_c|$. An *access node* is where end users reside and access content from, such as an ISP access network or point of presence. A potential *server node* represents a location that resources can be hired and a virtual server can be established, such as a server farm or server cluster. A *content source node* is where some streaming content is generated, for example a television or radio station. Directed edges in E represent potential network links offered by network providers, with $N_e = |E|$.

There are a total of K streaming channels. Each channel k originates from a source node $O(k) \in V_c$ and has bandwidth requirement γ_k . It is assumed that delivering channel k requires a storage (buffering) space of α_k at the streaming server, and each streaming session to clients will require a streaming bandwidth of γ_k and processing power of β_k , which is attributed to content enhancement/personalization functions. Service demands are modelled by the estimated number of concurrent sessions λ_{mk} required from each access node $m \in V_a$ for channel $k \in K$.

Establishing and delivering content from each server node i is assumed to incur the following costs (measured per unit time leased):

- c_i : server cost - which models the costs involved in establishing and maintaining a virtual server (e.g. resource allocation, shipping software, configuration, etc.)
- d_i : unit storage cost

- p_i : unit processing power cost
- b_{im} : unit streaming bandwidth cost between this server and access node m

Each server is assumed to have a maximum streaming capacity of B_i .

Note that streaming content is distributed to virtual servers using overlay multicast through backbone virtual links, while last mile proxy server-client streaming sessions are assumed to be created via Internet routed paths. To model possible restrictions on these client-server connections, each access node $m \in V_a$ is associated with a set $V_{s,m} \subset V_s$ of server locations to which its users can establish streaming sessions. This can be used to reflect either quality of service or administrative constraints on these Internet paths.

To establish the distribution back bone, virtual links must be purchased from network providers. Each link $(i, j) \in E$ is assumed to have:

- Link establishment cost e_{ij} , and
- Unit bandwidth cost b_{ij}

6.2.2 Decision Variables

To formulate this provisioning problem we define the following decision variables, which determines the location of virtual servers and virtual links:

$$y_i = \begin{cases} 1 & \text{if server } i \in V_s \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (6.1)$$

$$l_{ij} = \begin{cases} 1 & \text{if link } (i, j) | i, j \in V_c \cup V_s, i \neq j, \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (6.2)$$

In order to calculate the amounts of resources to be provisioned, we also need the following variables, which describe how each content channel is delivered through

the system:

$$x_{ik} = \begin{cases} 1 & \text{if channel } k \text{ is streamed to clients from server } i \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

$$t_{ij}^k = \begin{cases} 1 & \text{if multicast tree for } k \text{ includes link } (i, j) \\ 0 & \text{otherwise} \end{cases} \quad (6.4)$$

$$f_{ij}^{kn} = \begin{cases} 1 & \text{if content channel } k \text{ goes to server } n \in V_s \text{ via link } (i, j) \\ 0 & \text{otherwise} \end{cases} \quad (6.5)$$

$$r_{mi}^k \geq 0 \quad (6.6)$$

The continuous variable r_{im}^k shows the fraction of demand for content channel k from access node m that are directed to server i . The binary variable $f_{ij}^{kn} \in \{0, 1\}$ indicates if, in the distribution tree for channel k , the path from source $O(k)$ to a proxy server $n \in V_s$ goes through link $(i, j) \in E$. While not directly obvious from the problem description, this extra variable is needed to model the tree nature of the overlay multicast distribution. A complete list of these variables and problem parameters are included in Table 6.1 for easy reference.

6.2.3 Formulation

Using the above variables and parameters, the provisioning problem can be formulated as the following mixed integer linear programming model that minimizes total cost, including server and link initiation, storage, processing and bandwidth costs:

$$\begin{aligned}
 (\mathcal{S}) \quad \text{Min} \quad & \underbrace{\sum_{i \in V_s} y_i c_i}_{\text{site cost}} + \underbrace{\sum_{i \in V_s} \sum_{k \in K} x_{ik} (d_i \alpha_k)}_{\text{storage cost}} + \underbrace{\sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (\beta_k p_i + \gamma_k b_{im})}_{\text{streaming processing and bandwidth cost}} \\
 & + \underbrace{\sum_{(i,j) \in E} \sum_{k \in K} t_{ij}^k (\gamma_k b_{ij})}_{\text{link bandwidth}} + \underbrace{\sum_{(i,j) \in E} l_{ij} e_{ij}}_{\text{link cost}}
 \end{aligned} \quad (6.7)$$

Table 6.1 Parameters and Variables for Streaming ODN Provisioning Model

Notation	Meaning
<i>Resource parameters</i>	
b_{im}	unit delivery bandwidth from server i to access node m
c_i	server initiation cost
d_i	server unit storage cost
p_i	server unit processing cost
B_i	server streaming capacity
N_a	number of access nodes, $N_a = V_a $
N_c	number of content source nodes, $N_c = V_c $
N_s	number of sever nodes, $N_s = V_s $
n, i, j	subscripts for server nodes
m	subscript for access nodes
e_{ij}	link initiation cost, for link $(i, j) \in E$
b_{ij}	unit bandwidth cost on link (i, j)
N_e	number of potential links $L = E $
<i>Content parameters</i>	
α_k	storage requirement of content channel k
γ_k	streaming bandwidth requirement
β_k	streaming processing requirement
λ_{mk}	access rate for channel k from access node m
k	subscript for content objects
<i>Decision variables</i>	
y_i	server establishment indicator, $y_i \in \{0, 1\}$
x_{ik}	streaming proxy replication indicator, $x_{ik} \in \{0, 1\}$
t_{ij}^k	inclusion of link (i, j) in multicast tree k , $t_{ij}^k \in \{0, 1\}$
f_{ij}^{kn}	inclusion of link (i, j) in distribution path of channel k from source to proxy n , $f_{ij}^{kn} \in \{0, 1\}$
r_{mi}^k	request redirection ratio, $r_{mi}^k \in [0, \lambda_{mk}]$

Subject to:

$$\sum_{i \in V_s} r_{mi}^k = \lambda_{mk}, \quad m \in V_a, k \in K \quad (6.8a)$$

$$r_{mi}^k = 0, \quad i \notin V_{s,m} \quad (6.8b)$$

$$\sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \gamma_k \leq B_i, \quad i \in V_s \quad (6.8c)$$

$$r_{mi}^k \leq x_{ik} \lambda_{mk}, \quad m \in V_a, i \in V_s, k \in K \quad (6.8d)$$

$$\sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} = x_{nk}, \quad i = O(k) \quad (6.9a)$$

$$\sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} = 0, \quad i \in V_s, i \neq n \quad (6.9b)$$

$$\sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} = -x_{nk}, \quad i = n \quad (6.9c)$$

$$f_{ij}^{kn} \leq x_{nk}, \quad (i,j) \in E, k \in K, d \in V_s \quad (6.10a)$$

$$f_{ij}^{kn} \leq t_{ij}^k, \quad (i,j) \in E, k \in K, d \in V_s \quad (6.10b)$$

$$t_{ij}^k \leq l_{ij}, \quad (i,j) \in E, k \in K \quad (6.10c)$$

$$l_{ij} \leq y_i, \quad (i,j) \in E \quad (6.10d)$$

$$l_{ij} \leq y_j, \quad (i,j) \in E \quad (6.10e)$$

In this formulation, constraint (6.8a) makes sure that all service demand is taken into account, (6.8b) ensures that users in each access node m can only establish streaming sessions to the servers specified in $V_{s,m}$. Constraint (6.8c) enforces the streaming capacities of servers and (6.8d) means a content channel can only be streamed from servers sites chosen to be streaming servers for that channel. Constraints (6.9a), (6.9b) and (6.9c) ensure that for each channel, there exists a transmission path from its source, possibly via other servers, to each of its streaming servers. By considering f_{ij}^{kn} as the amount of flow from the source node to destination node n via link (i,j) , (6.9a), (6.9b) and (6.9c) can be seen as flow conservation constraints at the source, transit and destination nodes, respectively. This modelling technique is adopted from the flow formulation of Steiner multicast tree problems often seen in operations research literature (Zosin and Khuller, 2002).

The remaining constraints maintain logical interdependencies among variables: (6.10a) means a channel's transmission path to a proxy server is not needed if the channel is not streamed from that particular server; (6.10b) means the transmission path defined by variable f_{ij}^{kn} must be within the channel's multicast tree, which is defined by t_{ij}^k ; (6.10c) means that a multicast tree can only traverse a link if that virtual link is already established; finally, (6.10d) and (6.10e) mean that virtual servers must be established at both ends of a virtual link.

6.2.4 Problem Complexity

Compared to problems considered in previous chapters, the inclusion of link and path parameters in this provisioning problem has significantly increased its complexity, in terms of both variable and constraint numbers:

- Binary variables: $N_s + N_s K + L + LK + LKN_s$, i.e. $O(N_s^3 K)$, as $O(L) = O(N_s^2)$
Continuous variables: $N_a N_s K$, i.e. $O(N_a N_s K)$
Total number of variables is thus in the order of $N_s^3 K$
- Constraints: $N_a K + 2N_a N_s K + N_s + N_s K(2 + N_s) + 2LKN_s + LK + 2L$, i.e. $O(N_s^3 K)$

This provisioning problem is again an NP-hard problem, as in certain special situations it could be reduced to an instance of the classic NP-hard facility location problem (Sridharan, 1995), as follows: Consider the problem when there is only one content channel, no QoS constraints, graph G is a fully meshed network (thus any server location is a potential streaming proxy), and link initiation, bandwidth and server storage costs are all zero (i.e. $e_{ij} = 0$, $b_{ij} = 0$, $d_i = 0$). By eliminating redundant objective function components and constraints, dropping the unnecessary subscript k , and replacing $r_{mi} = u_{mi} * \lambda_m$, $u_{mi} \in \{0, 1\}$, the problem can be rewritten as:

$$\text{Min} \sum_{i \in V_s} y_i c_i + \sum_{i \in V_s} \sum_{m \in V_a} u_{mi} (\lambda_m \beta_k p_i)$$

Subject to:

$$\begin{aligned}
\sum_{i \in V_s} u_{mi} &= 1 \\
\sum_{m \in V_a} r_{mi} \gamma &\leq B_i, \quad i \in V_s \\
u_{mi} &\leq y_i, \quad m \in V_a, i \in V_s \\
u_{mi}, y_i &\in \{0, 1\}
\end{aligned}$$

which is a facility location problem, similar to the reduced problem we have seen in Section 4.2.4. Therefore, the resource provisioning problem for a live streaming ODN is also an NP-hard problem.

6.3 Heuristic Procedure

In this section we will develop a heuristic procedure based on Lagrangian relaxation in order to find near optimal solutions by solving the ODN provisioning problem above within reasonable computation time. Instead of the original hard problem, this heuristic will solve a series of simplified relaxed problems iteratively.

6.3.1 Applying Lagrangian Relaxation

The proposed Lagrangian heuristic simplifies the provisioning problem by relaxing constraints (6.8d) and (6.10b). Relaxing (6.8d) means that content requests are now allowed to be routed to locations where the required content is not available or a server is even not established, while relaxing (6.10b) means that the transmission path from a content source to a server does not have to traverse only the links within that content channel's distribution tree. These relaxations make the problem significantly easier to solve, as we will see in the next section, and the discrepancies caused will be rectified when a feasible solution is reconstructed, as in Section 6.3.3.

In relaxing these two sets of constraints the following terms are added to the objective

function of the original problem:

$$\sum_{m \in V_a} \sum_{i \in V_s} \sum_{k \in K} \mu_{mi}^k (r_{mi}^k - x_{ik} \lambda_{mk}), \text{ and}$$

$$\sum_{(i,j) \in E} \sum_{n \in V_s} \sum_{k \in K} w_{ij}^{kn} (f_{ij}^{kn} - t_{ij}^k)$$

where $\mu_{mi}^k, w_{ij}^{kn} \in \mathbb{R} | \mu_{mi}^k, w_{ij}^{kn} \geq 0$ are Lagrangian multipliers.

After re-arranging the objective function, the relaxed optimization problem becomes:

$$\begin{aligned} (S_{LR}) \quad \text{Min} \quad & \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (\beta_k p_i + \gamma_k b_{im} + \mu_{mi}^k) + \\ & + \sum_{i \in V_s} \sum_{k \in K} x_{ik} (d_i \alpha_k - \sum_{m \in V_a} \mu_{mi}^k \lambda_{mk}) + \sum_{(i,j) \in E} \sum_{k \in K} \sum_{n \in V_s} w_{ij}^{kn} f_{ij}^{kn} + \\ & + \sum_{(i,j) \in E} l_{ij} e_{ij} + \sum_{i \in V_s} y_i c_i + \sum_{(i,j) \in E} \sum_{k \in K} t_{ij}^k (\gamma_k b_{ij} - \sum_{n \in V_s} w_{ij}^{kn}) \end{aligned} \quad (6.12)$$

Subject to: (6.8a) to (6.10e).

The relaxed problem now can be decomposed into the following three independent subproblems

1. First subproblem:

$$(S_1) \quad \text{Min} \quad \sum_{i \in V_s} \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k (\beta_k p_i + \gamma_k b_{im} + \mu_{mi}^k) \quad (6.13)$$

subject to:

$$\begin{aligned} \sum_{i \in V_s} r_{mi}^k &= \lambda_{mk}, \quad m \in V_a, k \in K \\ r_{mi}^k &= 0, \quad i \notin V_{s,m} \\ \sum_{m \in V_a} \sum_{k \in K} r_{mi}^k \gamma_k &\leq B_i, \quad i \in V_s \end{aligned}$$

This subproblem involves request routing variables (r_{mi}^k) and is an instance of the classic transport problem, which can be solved efficiently by existing linear programming solution methods. In our work, the linear programming solver provided with optimization package Cplex is used to solve this subproblem.

2. Second subproblem:

$$(S_2) \text{ Min } \sum_{i \in V_s} \sum_{k \in K} x_{ik} \underbrace{\left(d_i \alpha_k - \sum_{m \in V_a} \mu_{mi}^k \lambda_{mk} \right)}_{A_{ik}} + \sum_{(i,j) \in E} \sum_{k \in K} \sum_{n \in V_s} w_{ij}^{kn} f_{ij}^{kn} \quad (6.15)$$

subject to:

$$\begin{aligned} \sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} &= x_{nk}, \quad i = O(k) \\ \sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} &= 0, \quad i \in V_s, i \neq n \\ \sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} &= -x_{nk}, \quad i = n \\ f_{ij}^{kn} &\leq x_{nk}, \quad (i,j) \in E, k \in K, d \in V_s \end{aligned}$$

This subproblem essentially determines the distribution path (specified by variables f_{ij}^{kn}) for each content channel k if it were to be streamed to users from server i .

3. Third subproblem:

$$(S_3) \text{ Min } \sum_{(i,j) \in E} l_{ij} c_{ij} + \sum_{i \in V_s} y_i c_i + \sum_{(i,j) \in E} \sum_{k \in K} t_{ij}^k (\gamma_k b_{ij} - \sum_{n \in V_s} w_{ij}^{nk}) \quad (6.17)$$

subject to:

$$\begin{aligned} t_{ij}^k &\leq l_{ij}, \quad (i,j) \in E, k \in K \\ l_{ij} &\leq y_i, \quad (i,j) \in E \\ l_{ij} &\leq y_j, \quad (i,j) \in E \\ t_{ij}^k, l_{ij}, y_i &\in \{0, 1\} \end{aligned}$$

This third subproblem, on the other hand, determines the locations of virtual servers and links.

6.3.2 Solving the Subproblems

All of the three subproblems resulting from relaxing the original model can be solved efficiently. The first subproblem, (S_1) , is a linear programming problem with only continuous variables and can be solved efficiently. In our heuristic implementation optimization package Cplex is used to solve this problem.

The second subproblem, (S_2) , can be further separated into $N_s \times K$ independent problems, one for each content channel - server location pair. The resulted subproblem for content channel k , server location n is:

$$\text{Min } x_{nk} A_{nk} + \sum_{(i,j) \in E} w_{ij}^{kn} f_{ij}^{kn}$$

subject to:

$$\begin{aligned} \sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} &= x_{nk}, \quad i = O(k) \\ \sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} &= 0, \quad i \in V_s, i \neq n \\ \sum_{j|(j,i) \in E} f_{ji}^{kn} - \sum_{j|(i,j) \in E} f_{ij}^{kn} &= -x_{nk}, \quad i = n \\ f_{ij}^{kn} &\leq x_{nk}, \quad (i,j) \in E, k \in K, d \in V_s \end{aligned}$$

where A_{nk} is as defined in equation (6.15). Note that we have changed the server location subscript from i to n in the $\sum_{i \in V_s} \sum_{k \in K} x_{ik} A_{ik}$ component of the objective function (6.15) before performing the above decomposition.

Leaving the $x_{nk} A_{nk}$ component of the objective function aside, the subproblem above becomes a shortest path problem, where the shortest path is to be found from the source node $O(k)$ to server $n \in V_s$, with w_{ij}^{kn} as edge weight for link (i,j) . Thus the above problem can be solved by solving this shortest path problem first, and then consider the following possibilities:

- if $A_{nk} + SP(n, k) \leq 0$, where $SP(n, k)$ is the total path weight as found by the shortest path problem, then $x_{nk} = 1$
- otherwise, $x_{nk} = 0, f_{ij}^{kn} = 0, \forall (i,j) \in E$

In subproblem (S_3) , as $t_{ij}^k \leq l_{ij}$, we have the following:

$$t_{ij}^k = \begin{cases} 1 & \text{if } l_{ij} = 1 \text{ and } (\gamma_k b_{ij} - \sum_{n \in V_s} w_{ij}^{nk}) \leq 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.20)$$

Therefore, in this subproblem, establishing link (i, j) (i.e. $l_{ij} = 1$), would contribute to the objective function an amount of $D_{ij} = e_{ij} + \sum_k \text{Min}\{0, (\gamma_k b_{ij} - \sum_{n \in V_s} w_{ij}^{nk})\}$. If D_{ij} , as defined above, is bigger than zero, then link (i, j) would not be established in the optimal solution to (S_3) and thus does not need to be considered. This is because if a link (i, j) with $D_{ij} > 0$ does get established in the optimal solution, simply deselecting this link (i.e. setting l_{ij} and associated t_{ij}^k variables to zero) would even further improve this supposedly optimal solution, which cannot happen. As a result, with $E^\#$ defined as the set of edges for which D_{ij} is non positive ($E^\# = \{(i, j) \in E | D_{ij} \leq 0\}$), problem (S_3) can be transformed into:

$$(S_{3eq}) \text{ Min } \sum_{(i,j) \in E^\#} l_{ij} D_{ij} + \sum_{i \in V_s} y_i c_i \quad (6.21)$$

subject to:

$$\begin{aligned} l_{ij} &\leq y_i, \quad (i, j) \in E^\# \\ l_{ij} &\leq y_j, \quad (i, j) \in E^\# \\ l_{ij}, y_i &\in \{0, 1\} \end{aligned}$$

Fortunately, we have been able to prove that due to its special form, the optimal solution to this problem can be found by replacing the binary variables with continuous variables ($y'_i, l'_{ij} \in [0, 1]$), solving the resulting linear relaxation form of the problem and finally rounding the the optimal solutions back into binary values. The detailed mathematical proof for this is presented in Appendix B.2.

Solving the three subproblems (S_1) , (S_2) and (S_3) separately gives a lower bound on the optimal objective function value of the original problem, (S) . Their solution also provides a basis from which a candidate feasible solution is constructed, as will be described in the next section.

6.3.3 Constructing a Feasible Solution

As a number of constraints have been relaxed, simply piecing together the subproblem solutions would not create a feasible solution to the original problem. For example, solving (S_1) may route content requests to locations where the requested content is not present, in solution to (S_2) . Similarly, solution to (S_2) may specify distribution paths that traverse links and nodes that are not established, according to (S_3) . In order to obtain a candidate feasible solution from solutions of the subproblems, such constraint violations need to be rectified.

The solution recovery process employed in this heuristic starts with request routing parameters obtained from solving subproblem (S_1) and goes on to establish necessary servers, network links and content distribution paths (i.e. overlay multicast trees) to make the solution feasible. This is because request routing parameters (variables r_{mi}^k) as obtained by solving (S_1) have already satisfied server resource capacity constraint (equation 6.8c). Thus by taking these parameters and establishing necessary distribution paths from content sources to indicated proxies, a feasible solution could be achieved.

A number of different solution reconstruction approaches were examined. The first approach, which we denote the *backbone trees* method, involves first establishing a backbone network connecting content sources and streaming proxies and then creating overlay distribution trees on this backbone. This method consists of the following main steps:

1. *Establishing streaming servers:* Using request routing information provided by solving (S_1) , streaming proxy locations are determined and virtual servers are established as necessary:

$$x_{ik} = y_i = 1, \text{ if } r_{mi}^k > 0$$

2. *Establishing backbone network:* In this second step, virtual links and possibly extra virtual servers have to be placed to create a backbone over which live content could reach the streaming proxies. This is done by starting with an initial set of servers and links as the starting backbone, and gradually growing this set (i.e. adding new links and servers) so that all streaming proxies are

reachable from their content sources. New servers and links are added by searching for the lowest-cost path, in terms of new server and link costs, from content source nodes to proxy locations that are yet reachable.

A number of slightly different variations of this step were experimented with, including:

- Starting with the initial set of links and servers selected in the solution to subproblem (S_3), the backbone network is then expanded if necessary by gradually adding other links and servers. The intuition is that although the subproblems are solved independently, they in fact influence each other via the Lagrangian multipliers, thus the links and servers selected subproblem (S_3) is probably a good starting point in looking for backbone links and servers. We denote this variation *backbone trees 1*.
 - Starting with the initial set of both links and servers selected in the solutions to (S_2) and (S_3). We denote this variation *backbone trees 2*.
 - Starting with links and servers selected in solution to (S_3), the backbone network is again gradually expanded by adding new servers and links if necessary, but this time with preference given to those already used by the solution to (S_2). This is done by reducing their the initiation costs of these links by half when they are considered to be added to be exiting backbone. We denote this variation *backbone trees 3*.
3. *Building multicast trees*: Once a backbone network connecting content sources and streaming proxies has been established, overlay multicast routes are established to distribute content to the streaming servers. Mathematically, constructing optimal multicast routes from a content source to a set of servers is equivalent to solving the classic Steiner tree problem (Ramanathan, 1996), which is also an NP-hard problem. However, in operations research literature there exist a number of efficient heuristics for this problem. In this work we have adopted the successive shortest path heuristic (F. K. Hwang and Winter, 1992), which seems to be the most suitable candidate to apply to a directed graph, to create multicast trees over an established backbone. This heuristic starts from a content source and gradually extends the distribution tree by adding the

shortest path, from any servers already traversed by the tree, to a server still outside the tree.

4. *Backbone pruning*: If some servers or links, although established in Step 2, end up unused in any of the distribution trees, they are removed in this final step. This pruning step has been added as experimental results show that the heuristic solutions are occasionally improved with its inclusion.

A simpler alternative solution construction method was also examined, which does not employ the Steiner tree problem to establish distribution trees. Instead, the distribution path for each content channel from the source to a proxy server is taken directly from the results of the shortest path subproblems of problem (S_2) . All backbone servers and links traversed by any of these paths are then established to make the solution feasible. This method is denoted the *simple paths* method.

Again, note that the objective function value given by a reconstructed feasible solution is always an upper bound of the original problem's optimal objective function value. Therefore, the gap between this upper bound and the lower bound obtained by solving the relaxed problem reflects the worst case distance of the solution from the optimal, which can be used to ascertain the heuristic solution quality.

6.3.4 Subgradient Procedure

As can be seen in the previous sections, with each value of the Lagrangian multiplier vectors $(\mu$ and $w)$, a feasible solution can be reconstructed, whose quality can be measured by the gap between its objective function value and the lower bound. Our heuristic again uses the common subgradient procedure to adjust these multipliers in a way to reduce the upper bound - lower bound gap, and thereby improving heuristic solution quality, between iterations.

The mathematical formulas used to adjust Lagrangian multipliers in this case are shown in Appendix A.4 and more detailed background information on the subgradient procedure can be found in (Glover and Laguna, 1993). The numerical results obtained by this heuristic will be presented and discussed at the end of this chapter, in Section 6.5.

6.4 Online Optimization

In this section we explore the use of a separate online optimization process to fine-tune service parameters in order to accommodate short term demand fluctuations, given an established virtual topology. This process is intended to be used over short time scales, for which re-provisioning already allocated resources is not practical, and may involve redirecting/rebalancing service demand among servers or reorganizing content distribution trees.

In order to model this process, the following parameters and variables are defined:

The established overlay network is modelled as \tilde{G} , which is a subgraph of the original network, $\tilde{G} = \langle \tilde{V}, \tilde{E} \rangle \subset G$, where $\tilde{V}_s = \tilde{V} \cap V_s$ is the set of established virtual servers and \tilde{E} is the set of established links. Each virtual server $i \in \tilde{V}_s$ has an allocated streaming capacity of \tilde{B}_i , storage space of \tilde{D}_i and processing power of \tilde{P}_i . Each virtual link $(i, j) \in \tilde{E}$ has bandwidth of \tilde{W}_{ij} .

A utility value ρ_{mk} is associated with each unit of demand for channel k from access node m . This value is content and location dependent and thus can be used to prioritize service and/or customer classes.

Assuming that current service demand is λ_{mk}^{\sim} , which might be different from initially estimated values, the aim of the online optimization process would be to adjust distribution parameters so that total service utility is maximized, given the constraints of currently allocated resources. Distribution parameters to be determined by this process are denoted by adding the *tilde* (\sim) notation to corresponding variables defined earlier in (6.3), (6.4), (6.6) and (6.5). Thus we now have the following new variables: \tilde{x}_{ik} , \tilde{t}_{ij}^k , \tilde{r}_{mi}^k and \tilde{f}_{ij}^{nk} .

We have examined a number of alternative online optimization models, including:

1. *Full optimization*: This is a centralized optimization that takes into account all service adjustment possibilities, including rerouting content requests, relocating streaming proxies and rerouting content distribution trees. The problem is formulated as follows:

$$(T_1) = \text{Max} \sum_{m \in V_a} \sum_{i \in \tilde{V}_s} \sum_{k \in K} \tilde{r}_{mi}^k \rho_{mk} \quad (6.23)$$

Subject to:

$$\sum_{i \in \tilde{V}_s} \tilde{r}_{mi}^k \leq \tilde{\lambda}_{mk} \quad (6.24a)$$

$$\tilde{r}_{mi}^k = 0, \quad i \notin V_{s,m} \quad (6.24b)$$

$$\tilde{r}_{mi}^k \leq \tilde{x}_{ik} \tilde{\lambda}_{mk}, \quad m \in V_a, i \in \tilde{V}_s, k \in K \quad (6.24c)$$

$$\sum_{j|(j,i) \in \tilde{E}} \tilde{f}_{ji}^{kn} - \sum_{j|(i,j) \in \tilde{E}} \tilde{f}_{ij}^{kn} = \tilde{x}_{nk}, \quad i = O(k) \quad (6.25a)$$

$$\sum_{j|(j,i) \in \tilde{E}} \tilde{f}_{ji}^{kn} - \sum_{j|(i,j) \in \tilde{E}} \tilde{f}_{ij}^{kn} = 0, \quad i \in \tilde{V}_s, i \neq n \quad (6.25b)$$

$$\sum_{j|(j,i) \in \tilde{E}} \tilde{f}_{ji}^{kn} - \sum_{j|(i,j) \in \tilde{E}} \tilde{f}_{ij}^{kn} = -\tilde{x}_{nk}, \quad i = n \quad (6.25c)$$

$$\tilde{f}_{ij}^{kn} \leq \tilde{x}_{nk}, \quad (i, j) \in \tilde{E}, k \in K, n \in \tilde{V}_s \quad (6.26a)$$

$$\tilde{f}_{ij}^{kn} \leq \tilde{t}_{ij}^k, \quad (i, j) \in \tilde{E}, k \in K, n \in \tilde{V}_s \quad (6.26b)$$

$$\sum_k \tilde{t}_{ij}^k b_k \leq \tilde{W}_{ij} \forall (i, j) \in \tilde{E} \quad (6.26c)$$

$$\sum_{m \in V_a} \sum_{k \in K} \tilde{r}_{mi}^k \gamma_k \leq \tilde{B}_i, \quad i \in \tilde{V}_s \quad (6.26d)$$

$$\sum_{m \in V_a} \sum_{k \in K} \tilde{r}_{mi}^k \beta_k \leq \tilde{P}_i, \quad i \in \tilde{V}_s \quad (6.26e)$$

$$\sum_{k \in K} \tilde{x}_{ik} \alpha_k \leq \tilde{D}_i, \quad i \in \tilde{V}_s \quad (6.26f)$$

In this formulation, constraint (6.24a) ensures the demands assigned to servers cannot exceed the total offered amounts. Constraint (6.26c) means bandwidth consumption cannot exceed allocated amount \tilde{W}_{ij} on each virtual link (i, j) . Constraints (6.26d), (6.26e) and (6.26f) keep resource consumption at each server at or below the allocated capacities, for streaming bandwidth, computation and storage space, respectively. Equations (6.25a), (6.25b) and (6.25c) are flow conservation constraints, which ensure that there exists a distribution path from each content source to each streaming proxy that delivers its content. These are similar to constraints (6.9a), (6.9b), (6.9c) in provisioning model (S) (6.7).

2. *Partial optimization:* This is a simplified centralized optimization where content request routing is reorganized to minimize server overloads, but proxy locations and content distribution paths are not changed. The problem formulation thus has the following form:

$$(T_2) = \text{Max} \sum_{m \in V_a} \sum_{i \in \tilde{V}_s} \sum_{k \in K} \tilde{r}_{mi}^k \rho_{mk} \quad (6.27)$$

Subject to:

$$\sum_{i \in \tilde{V}_s} \tilde{r}_{mi}^k \leq \lambda_{mk} \quad (6.28a)$$

$$\tilde{r}_{mi}^k = 0, \quad i \notin V_{s,m} \quad (6.28b)$$

$$\tilde{r}_{mi}^k \leq x_{ik} \lambda_{mk}, \quad m \in V_a, i \in \tilde{V}_s, k \in K \quad (6.28c)$$

$$\sum_{m \in V_a} \sum_{k \in K} \tilde{r}_{mi}^k \gamma_k \leq \tilde{B}_i, \quad i \in \tilde{V}_s \quad (6.28d)$$

where streaming proxy locations, as defined by x_{ik} are known and fixed.

Comparing to the full optimization, this problem is much simpler and much easier to solve.

3. *Local optimization:* This is the simplest approach, where instead of a global optimization each server independently performs a local optimization that selects which requests to honour, in order to maximize service utility, if the offered load is greater than its capacity. At ODN server i , given that the amount of requests for channel k from access node m is r_m^k , the local optimization can be formulated as follows:

$$(T_3) = \text{Max} \sum_{m \in V_a} \sum_{k \in K} \tilde{r}_m^k \rho_{mk} \quad (6.29)$$

Subject to:

$$\tilde{r}_m^k \leq r_m^k \quad (6.30a)$$

$$\sum_{m \in V_a} \sum_{k \in K} \tilde{r}_m^k \gamma_k \leq \tilde{B} \quad (6.30b)$$

Note that in this model the unnecessary subscript i has been dropped.

4. *No optimization:* As a comparison benchmark, we also examined the situation where no optimization is carried out. Instead, content requests are assumed to arrive randomly and handled on a first come first serve basis at each server.

By studying these different approaches we aim to compare their performance and also to evaluate the usefulness of the online optimization concept, in other words, how much performance improvement is achieved by applying online optimization. The study is also expected to give insight into the impacts of demand estimation inaccuracy on ODN topology quality.

6.5 Numerical Results

The optimization models and heuristics developed in this chapter have been implemented in *C++*, using Cplex libraries (Cplex, 2001) to solve linear programming subproblems and the Graph Template Library (GTL, 2001) to model graph structures and solve shortest path problems. The implementation has been numerically evaluated using network topologies generated by the GT-ITM topology generator (Calvert et al., 1997), and randomly generated cost parameters.

All computation presented in this chapter was done using machines with 2.4GHz CPU's and 1GB RAM.

6.5.1 Provisioning Model Behavior

The output of the resource provisioning model (S) was observed on problems with varying resource cost parameters and it was found that the number of servers to be established in a topology generally varies with the relative significance between transport costs (i.e. link and bandwidth costs) and server costs (site and server resource costs). As an example, Figure 6.1 shows the optimal overlay topologies obtained in problems with a 20 node network. In this example, bandwidth costs are initially significantly higher than server costs, and each access node generates the same amount of session requests. Only one content channel is considered in the case shown in this figure for the sake of example clarity. The resulting optimal topology is shown in Figure 6.1(a), where dark color links and nodes indicate established virtual servers

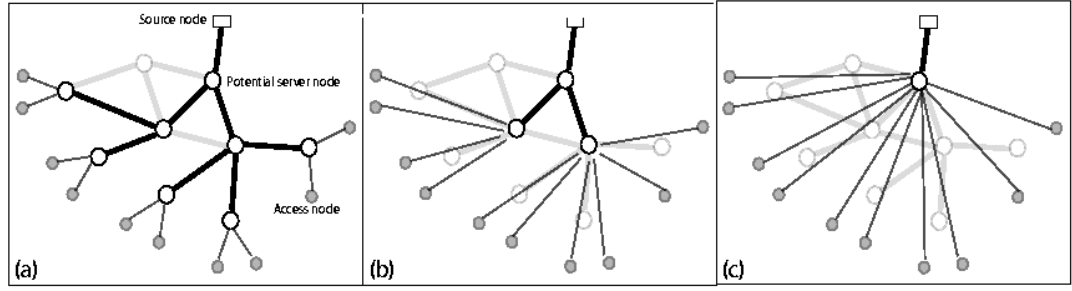


Figure 6.1 Overlay topology changes in reaction to resource cost variations: **(a)** high bandwidth and low server cost **(b)** server cost $\times 10$ **(c)** server cost $\times 20$

and backbone links. It can be seen that when transport costs are high, content is pushed to close to end-users in order to minimize transport cost by taking advantage of backbone overlay multicast capability. As server costs are raised, the model reacts by establishing fewer and fewer servers, as seen in Figures 6.1(b) and 6.1(c). Similar behavior is also observed with larger problems.

It was seen that raising server costs by either raising site cost or storage resource cost results in the same effect described above. Interestingly, raising server computation power costs does not have the same effect. This can be explained by the fact that computation requirement (c_k) is modeled as per streaming session, thus server proximity and server number has no effect on the total computation power required. However, computation cost variation between servers does effect the choice of server location, just as any other parameters. Besides resource cost, service demand also affects the optimal topology. Obviously, with lower demand, streaming bandwidth costs would reduce and so does the cost saving from overlay multicast, which would then lead to similar topology changes.

6.5.2 Heuristic Performance

This section aims to compare the performance of feasible solution reconstruction methods described in Section 6.3.3, as well as the overall performance of the Lagrangian heuristic developed.

Numerical results obtained from a large number of experiments showed that the performance of the construction methods based on Steiner trees (namely *backbone*

trees 1, *backbone trees 2* and *backbone trees 3*) are relatively close, while that of the simpler *simpler paths* method is significantly worse. As these procedures differ in the construction of backbone servers, links and content distribution trees, such results suggest that simply adopting distribution paths given by subproblem (S_2) without using tree heuristics would probably establish more paths than necessary, thus incurring inefficient backbone servers, links and bandwidth usage.

This is illustrated in Figure 6.2, where results obtained from a series of related problems are compared. The problems were generated from the same base problem by multiplying link/bandwidth cost with varying multipliers between 0.1 to 20. This is because higher cost is expected to result in lower quality solutions if there are resource inefficiencies. The experiment is also repeated with problems based on three different graphs with the same number of nodes but different link density, in order to observe its effects on heuristic accuracy.

In Figure 6.2, the objective function values of heuristic solutions obtained with different solution construction methods are plotted together against the link cost multiplier. As the objective function values are normalized by a common lower bound for each problem, the closer a normalized objective is to 1, the better the heuristic is.

All plots in this figure show a common trend, where higher link and bandwidth costs lead to lower solution quality for the *simple paths* method, while the tree based methods achieve markedly better solutions and remain consistently efficient as link cost increases. The gap between *simple paths* solution and tree-based solutions also increases significantly faster at higher graph density, as can be seen by comparing plots in Figures 6.2.a, 6.2.b and 6.2.c. This is because at higher graph density, there are more possible paths between a content source and proxy server location, thus efficient path selection would have greater effects on solution quality.

Interestingly, the same behavior is not observed when server cost is increased. As can be seen from Figure 6.3, which compares results obtained with different procedures when server cost is varied, the quality of solutions by *simple paths* procedure seems to improve at higher server costs. This may be because although inefficiencies in establishing distribution paths may lead to unnecessary server establishments, increasing server cost also affects the overall optimization process, which establishes fewer proxy locations. This results in fewer transmission paths to be created, which

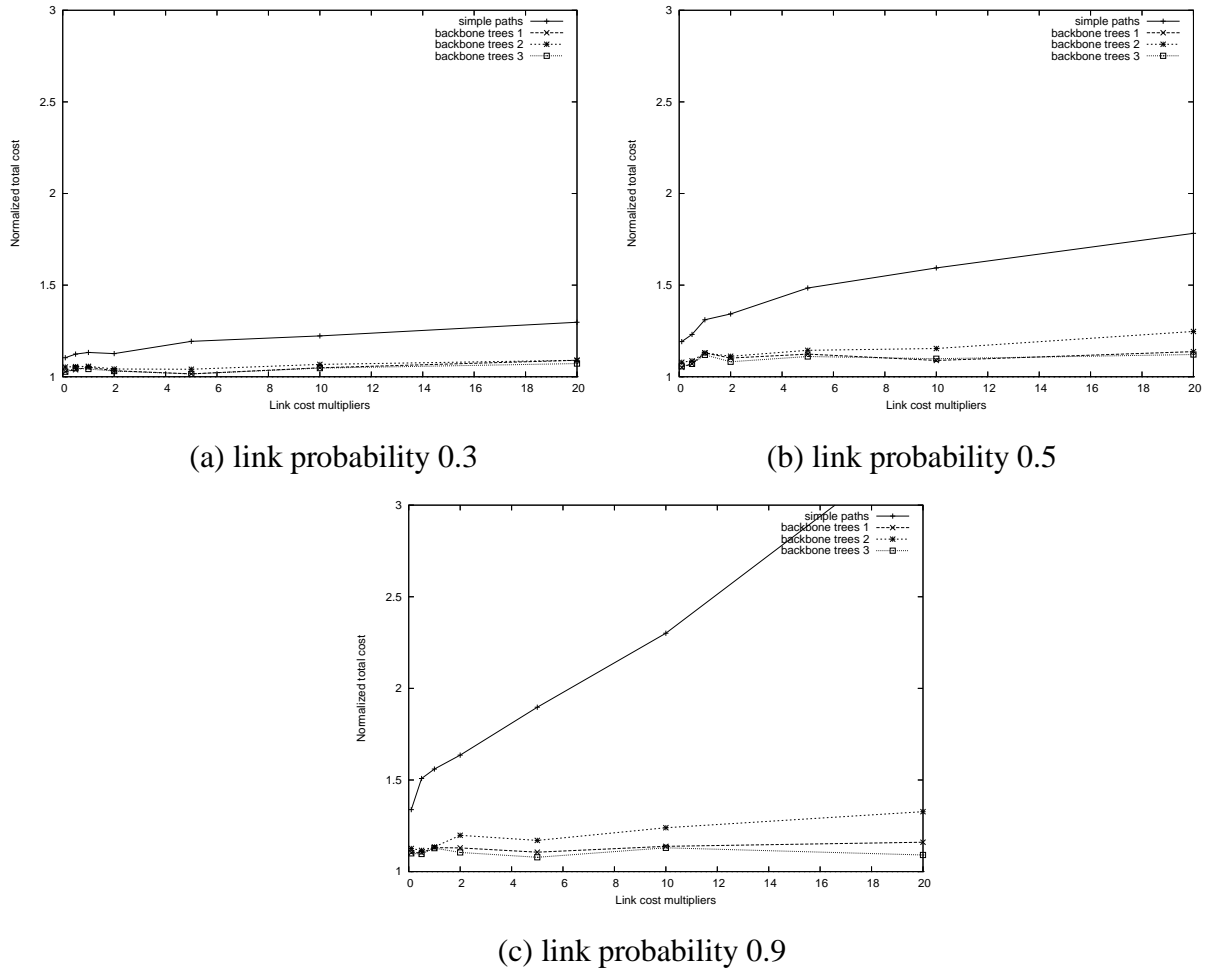


Figure 6.2 Normalized heuristic solutions obtained with different solution construction methods plotted against link cost multipliers, for 60-node graphs with different link density

in turn reduces the impact of inefficient path selection procedures. This is illustrated in Figure 6.4, where the normalized objective function value obtained with the *simple paths* procedure is plotted together with the total number of proxy servers as a percentage of available servers. Note that the number of servers had to be normalized by the total number of available servers in order to place the two graphs in the same plot. This figure appears to suggest that a certain correlation exists between the number of proxy servers (and thus number of paths required) and heuristic solution quality achieved by the *simple paths* procedure.

Experimental results, as seen in Figures 6.2 and 6.3, also showed that amongst the different tree based solution recovery procedures, *backbone trees 1* and *backbone*

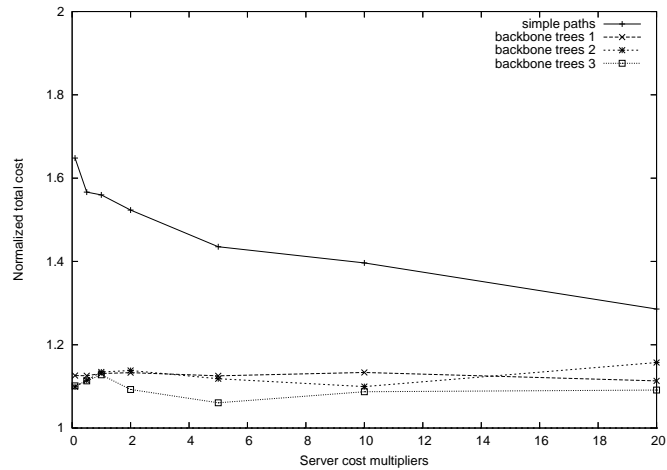


Figure 6.3 Normalized heuristic solutions obtained with different solution construction methods plotted against server cost multipliers, in a graph with link probability 0.9

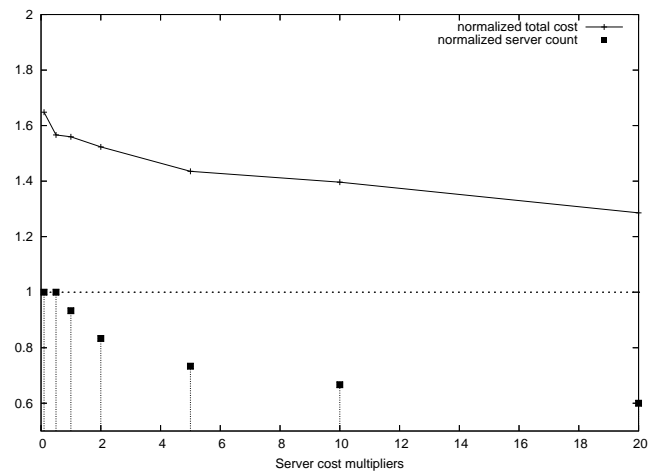


Figure 6.4 Normalized heuristic solution obtained with *simple paths* procedure, with normalized number of established proxies super-imposed

trees 2, which establish the backbone network by initially assuming servers and links indicated by subproblem (S_3) only, are very close and have consistently the best performance. Therefore, in the rest of this chapter, heuristic results presented are obtained with either of these methods.

More detailed numerical results obtained from a large range of problems, with different graph sizes and resource costs, are included in Table 6.2. The results in this table is shown in terms of problem size parameters (namely N_s , N_a , N_c and K), objective function value C_S , lower bound LB , percentage of total costs by server initiation, storage, processing, bandwidth and link initiation (c_c , c_d , c_p , c_b and c_l , respectively), the *gap* between heuristic solution and lowerbound, and optimal result if obtainable with Cplex.

These results show that the heuristic is able to achieve reasonable accuracy, with a maximum gap under 20% from optimal solution. This performance is also seen to be consistent over a large range of different cost combinations.

In terms of computation time, the heuristic is able to handle reasonably large problems, while solving for the optimal solution with Cplex is only possible at very small problem sizes.

6.5.3 Online Optimization

This section aims to present and discuss numerical results from the online optimization models. The online optimization process is designed to fine-tune content distribution parameters so that an existing topology can be utilized most effectively as service demands deviate from initially estimated levels. Thus in evaluating online provisioning models, we seek answers to the following questions:

- How far service demand can deviate from estimated levels before the provisioned topology becomes inadequate.
- As demand fluctuates, how much improvement can be produced by adopting online optimization.
- Among the different online optimization algorithms, which offers the most practical solution in each situation.

The online optimization model was evaluated using a 30-node network, with 20 content channels whose utilities (ρ_{mk}) are randomly generated between 1 and 2. Service demand estimates (λ_{mk}) are also randomly generated. With these parameters, an

Table 6.2 Numerical Results for Streaming Content ODN Provisioning

Problem size					Heuristic result								Cplex result		
N_s	N_a	N_c	K	C_S	LB	$c_c(\%)$	$c_d(\%)$	$c_p(\%)$	$c_b(\%)$	$c_l(\%)$	$gap(\%)$	time	Optimal	$gap^*(\%)$	time
10	10	2	10	12030	11567	24	10	16	30	20	4.0	30s	11911	1.0	1.5m
				14022	12864	5	2	15	36	42	9.0	25s	13614	3.0	3.0m
				11439	10894	7	4	12	27	20	5.0	42s	10999	4.0	50s
				15439	14161	31	30	22	11	6	9.0	30s	14989	3.0	5.0m
15	15	5	20	29132	27745	21	22	13	26	18	5.0	2.5m	28283	3.0	16m
				24217	22633	38	16	24	20	2	7.0	1.5m	23064	5.0	55m
				22193	20361	10	22	41	18	9	9.0	2.0m	21339	4.0	50m
				42476	39330	2	6	7	43	42	8.0	3.0m	40072	6.0	60m
20	20	5	50	39056	36501	25	20	15	28	12	7.0	10m	-	-	-
				37238	32381	7	10	4	45	32	15	15m	-	-	-
				42139	39018	3	5	9	59	24	8.0	23m	-	-	-
				53012	50011	35	27	21	15	2	6.0	12m	-	-	-
30	30	10	100	47605	42401	41	12	24	16	7	11	3.0hr	-	-	-
				10732	10197	21	18	16	22	23	5.0	1.5hr	-	-	-
				73051	60632	15	19	22	33	21	17	1.0hr	-	-	-
				86123	76151	14	20	31	23	12	12	2.5hr	-	-	-
54	54	10	500	88150	81036	19	21	25	24	11	10	15hr	-	-	-
				79138	64893	30	26	18	16	10	18	12hr	-	-	-
				97354	86921	12	4	1	65	18	12	20hr	-	-	-
				53451	51395	40	18	21	11	10	4.0	17hr	-	-	-

overlay network topology is determined using the offline provisioning process.

To model demand variation, “actual” service demand ($\tilde{\lambda}_{mk}$) is then created by introducing fluctuations into the estimated values. Demand variation is specified and measured by the maximum deviation from estimated values, in percentage. For example, an $x\%$ deviation from estimated demand matrix λ_{mk} would be a matrix randomly generated between $\max\{(1 - x\%)\lambda_{mk}, 0\}$ and $(1 + x\%)\lambda_{mk}$. Note that this fluctuation would only affect the nonzero entries of the demand matrix and thus variations are created in demand levels, but not geographical locality.

We also experimented with variations that involves not only demand level fluctuations but also geographical shifts. This is modelled by generating initial demand in localized patterns (i.e. each content channel is requested by only a portion of access nodes), and then allowing up to $x\%$ of demand at each location to be shifted to other locations. Two types of geographical shifts are defined: *gradual shift*, where demand shifts are restricted to neighboring nodes only, and *random shifts*, where demand can be randomly reallocated to any other access nodes in the network. Figures 6.5(a), 6.5(c) and 6.5(b) plot the total service utility retained using online optimization against demand deviation when demand fluctuates without geographical shifts, with *gradual geographical shifts* and with *abrupt geographical shifts*, respectively. Total service utility achieved by each online optimization method is *normalized* by the maximum possible utility (i.e. $\sum_k \sum_m \lambda_{mk} \rho_{mk}$), which would be achieved if all demand was satisfied. Since this maximum utility varies between experiments, normalizing total utility values allows results from different scenarios to be plotted together and compared. Since many parameters in the experiment are randomly generated, the results presented here were obtained by repeating the experiment 5 times and taking average values.

As expected, the graphs show a clear and consistent order in the performance of the different online optimization algorithms, with the best performing being *full optimization*, then *partial optimization*, *local optimization* and doing no online optimization gives the worst performance. Performance improvements achieved by the using online optimization are significant, as indicated by the gap between results obtained by *no optimization* approach and others.

It can also be seen from the graphs that except when there are abrupt geographical

demand shifts, the differences between full and partial optimization results are not significant. These models can both retain above 90% total utility for demand deviations up to 40%, and maintain above 80% total utility for demand deviation as high as 60%. This means that partial optimization is a useful approach, as it is a much simpler problem and can be solved efficiently. Local optimization, on the other hand, can achieve comparable results for only low demand deviations (up to 20%).

When abrupt geographical demand shifts are introduced, however, full optimization performs markedly better than all other approaches. This is because significant deviations in both demand level and locality would require reconstruction of content distribution trees, which is performed only in the *full optimization* model. In these situations, while a full optimization can still maintain performance comparable to other situations, both partial and local optimizations become inefficient, with total utility dropping to well below 50% at high demand deviation levels.

These results suggest in situations where demand levels fluctuate slightly (below 20%) without abrupt geographical shifts, the simple local optimization approach is an adequate solution. As demand deviations increase beyond 20% mark, a global optimization that takes into account request rerouting and inter-server load balancing is necessary. However, if such high deviations are experienced together with abrupt shifts in demand locality, a full optimization that not only reroutes requests but also reroutes content distribution paths, is required.

Note that in this section, random demand deviations were assumed, thus an $x\%$ deviation does not necessary mean $x\%$ increase in total demand. Instead, it is more likely to be shifted within the network, among content channels or among user locations. In other words, online optimization is considered here in a context of random inaccuracies in demand estimates, not demand surges. In situations where estimated total demand level is decidedly lower than the actual one, online optimization would be of little use. Instead, either re-provisioning or over-provisioning would be required.

6.6 Conclusion

This chapter has considered the distribution of live streaming multimedia content using Overlay Distribution Networks and investigated its implications on the ODN

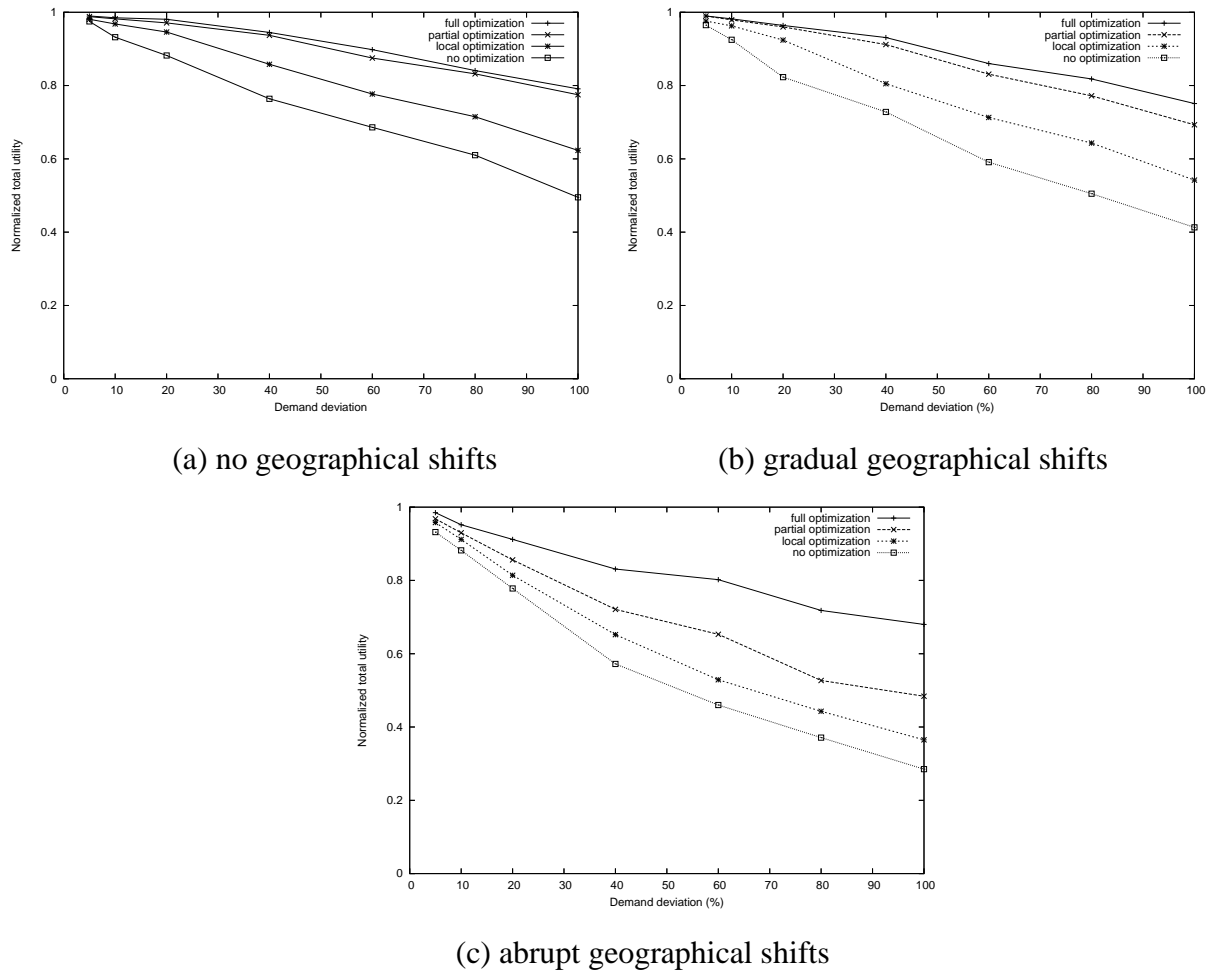


Figure 6.5 Normalized total service utility by different online optimization methods vs. demand variation created by different types of demand fluctuations

provisioning problem. With this class of applications, we envisage that virtual ODN servers would function not only as streaming proxies, but also content processing nodes for content adaptation/enhancement purposes and overlay multicast nodes. Streaming content is distributed to streaming servers from the live source using overlay multicast over the ODN backbone. As ODN provisioning in this case requires the consideration of not only front end servers but also backbone servers, links and content distribution trees, the problem is significantly transformed and becomes substantially different from those previously considered.

We have addressed this challenge by formulating an ODN provisioning model that determines not only the optimal ODN server and link configuration, but also how dis-

tribution trees are routed and how live content should be distributed from sources to servers. As this provisioning problem is an NP-hard optimization problem, we went on to develop a heuristic based on Lagrangian relaxation and subgradient optimization that could find near optimal solutions efficiently. Numerical results showed that the proposed heuristic is able to achieve promising solution accuracy (within 20%) for large problems with computation time in the order of hours.

We also proposed and studied a separate online optimization process that aims to adapt the provisioned network to demand fluctuations. A number of alternative online optimization models were developed and evaluated, including a *full optimization* and *partial optimization*, which are centralized processes, and a *local optimization*, which is a simple, distributed optimization to be carried out independently at each server.

Experimental results suggested that with small demand fluctuations, the simple local optimization is adequate, while with greater fluctuations the partial optimization can maintain good performance in most cases. Only in the presence of abrupt geographical demand shifts is a full optimization necessary.

Overall, using online optimization allows service performance to be maintained at above 80% of total utility for demand deviations of up to 60%. This indicates that the provisioning and online optimization models together form an efficient provisioning tool capable of creating optimal topologies that can also handle large variations in service demand.

Chapter 7

Improving ODN Provisioning Scalability with Content Clustering

7.1 Introduction

A potential limitation of the ODN provisioning models developed in this thesis is computation scalability. As seen in previous chapters, even when efficient heuristics are used, finding the desired ODN topology for problems with up to 1000 content items already requires computation time in the order of hours. In a provisioning problem with realistic dimensions, however, this number is likely to be significantly larger, for example in the order of 10,000s or 100,000s, which may require unacceptably long computation time.

Therefore, in this chapter we will explore the use of *content clustering* to further improve the scalability of our ODN provisioning models. With content clustering, multiple content items are grouped (i.e. *clustered*) together, with each group treated as a single item during the provisioning process, which effectively reduces the total number of items to be considered and therefore the provisioning complexity.

Clustering, however, may also lead to lower provisioning solution quality due to less efficient resource usage. For example, some items may get replicated (as part of a cluster) where it is not actually required or more server resources may get allocated than actually used. Work in this chapter therefore aims to develop efficient clustering methods that can achieve content item number reduction without significantly compromising solution quality.

Previously, content clustering in a CDN environment has been examined by (Chen et al., 2003). In this study, the authors aimed to improve the scalability of CDN content placement problems by grouping web pages into clusters, which are then replicated together. Clustering was done by defining a clustering distance between every pair of content items based on their attributes, and applying standard clustering algorithms, such as the classic *K-split* algorithm (Gsieniec et al., 2004), to group them. The study explored a number of different distance metrics, including spatial access patterns, temporal access patterns, temporal correlation of access patterns and content popularity (i.e. access frequency). Using trace driven simulations, it was found that clustering based on either popularity or spatial access distribution gives the best results, significantly reducing the placement problem complexity and thus computation time (down to a few percent of original problem in some cases) while still achieving comparable solution quality.

Work in this chapter, while sharing the same motivation and adopting the key concepts from this existing study, is different in several aspects. Firstly, the target optimization problems where content clustering is to be applied are set in the ODN environment and thus are substantially different, with a different set of objectives and constraints, as discussed earlier in Chapters 1 and 3. Therefore whether existing clustering methods are still applicable remains to be seen. Secondly, in ODN provisioning content items are characterized by more than just their spatial demand distribution. As will be demonstrated later in this chapter, other factors such as processing requirement, bandwidth requirement and revenue can also have great influence on the clustering efficiency and thus need to be taken into account. As a result, the content clustering problem becomes significantly more challenging and existing clustering techniques may need to be adapted accordingly.

In looking for a suitable content clustering approach, we propose and examine a hierarchical clustering method, in which content items are first grouped based on their resource requirement characteristics, and then clustered based on their spatial demand distribution and revenue levels. Experimental results, as will be presented in this chapter, show that this clustering approach has significant performance improvements when used in ODN provisioning compared to existing methods.

Throughout this thesis, ODN provisioning has been considered for a number of dif-

ferent content applications with different content types and provisioning models. However, the motivation and objective of the clustering process, as stated above, applies equally to all. Furthermore, in the provisioning models we have developed, content units are characterized in mathematically similar ways, with generic attributes such as demand patterns, storage, processing power and bandwidth requirements, which would require similar considerations by the clustering process. Therefore, we believe content clustering for the different ODN provisioning models would require similar techniques, and will examine them together in this chapter. Any differences that only apply to a particular model will be pointed out along the way.

The chapter is organized as follows. Section 7.2 further discusses general aspects of employing content clustering as part of the ODN provisioning process, including its integration, content cluster attribute calculation and general clustering techniques. Section 7.3 first presents an analysis of content attributes that may need to be considered by the clustering process. It then applies clustering methods used in previous studies into ODN provisioning scenarios and demonstrates that by failing to take into account new content attributes introduced in ODN provisioning models, such as revenue or resource consumption, the existing content clustering techniques have inadequate performance. In the remainder of Section 7.3, we propose and examine a number of modifications to existing content clustering methods in order to align the new content attributes with the clustering metric. Experimental results will then be presented to demonstrate that the proposed approach produces significant performance improvements. Finally, major conclusions from this study will be presented in Section 7.4.

7.2 General Clustering Framework

With content clustering, the ODN provisioning would now involve the following steps:

1. *Clustering*: Content items are grouped into clusters, which are then considered and replicated together as single items. Cluster attributes are calculated based on member items.

2. *Optimization*: ODN provisioning is performed on the population of clusters instead of original content items. The resulting output (i.e. set of variables x , y , r etc.) will be in terms of the cluster population.
3. *De-clustering the result*: In order to obtain a final solution in terms of the original content items, the optimization output must be translated. This is done by “breaking down the clusters”. In other words, server, link, replication locations and distribution paths of clusters are adopted for all of its members. Request routing parameters for individual items are also to be extracted from request routing information for their clusters. Therefore it is crucial for the clustering process to be done in a way that a solution in terms of clusters can always be translated into a *feasible* solution in terms of original individual items.

A key issue in designing the clustering process is how to calculate the attributes of a content cluster. The various attributes of a cluster, such as resource requirements, access frequency or revenue, must reflect the characteristics of the items it contains and also allow solutions in terms of clusters to be translated into solutions in terms of individual items. Therefore in our work the following scheme is used.

Assume that $K_{\bar{c}}$ is the set of content items assigned to cluster \bar{c} , then the attributes of this cluster are calculated based on its members:

- Storage requirement is the total requirements of member items, since they would be replicated together

$$\alpha_{\bar{c}} = \sum_{k \in K_{\bar{c}}} \alpha_k$$

- Access rate for the cluster is sum of requests for member items:

$$\lambda_{m\bar{c}} = \sum_{k \in K_{\bar{c}}} \lambda_{mk}$$

- Delivery computation and bandwidth requirements are the maximum requirements by member items:

$$\beta_{\bar{c}} = \max_{k \in K_{\bar{c}}} \{\beta_k\}$$

$$\gamma_{\bar{c}} = \max_{k \in K_{\bar{c}}} \{\gamma_k\}$$

The *max* criteria must be used here to ensure that a solution based on clusters can always be converted into a feasible solution for individual items. For example, if the optimization output directs a total amount r of requests for contents in cluster \bar{c} to a certain server, the amount may actually include any combination of requests for individual items, possibly with different resource requirements, in \bar{c} .

- Revenue of the cluster is the averaged revenue:

$$\rho_{\bar{c}} = \frac{\sum_m \sum_{k \in K_{\bar{c}}} \lambda_{mk} \rho_k}{\sum_m \sum_{k \in K_{\bar{c}}} \lambda_{mk}}$$

Note that depending on the particular provisioning scenario and the type of content items in question, such as web pages (provisioning model (4.4) in Chapter 4), pay-per-view content items (provisioning model (5.5) in Chapter 5), or live streaming multimedia channel (as in provisioning model (6.7) in Chapter 6), some attributes in this list may not apply. For example, content revenue attributes (ρ_k) are only used in the profit-based ODN provisioning model.

In grouping content items, the general approach used in the previous study (Chen et al., 2003) is adopted, where clustering distance metrics are defined between every pair of content items and standard clustering algorithms are then used to group them into clusters.

The results shown in (Chen et al., 2003) were obtained using the classic *K-split* clustering algorithm (Gsieniec et al., 2004), which minimizes the maximum diameter of all clusters while limiting the number of clusters. Apart from this algorithm we also implemented the classic *K-mean* algorithm (Mirkin, 1996), which minimizes the total distances from content items to their cluster centers, for a given number of clusters. However, no significant difference in performance between the two was observed and the results described in this chapter assumes the use of *K-mean* clustering algorithm.

7.3 Content Clustering in ODN Provisioning

7.3.1 Analysis

To efficiently cluster content items in an ODN provisioning process, we believe the following factors need to be taken into account:

- *Spatial distribution of demand*: Since content items in a cluster are to be considered and replicated together as a single item by the provisioning process, clustering items with distinctly different spatial demand distribution patterns would result in unnecessary replication, which leads to wasted resources.
- *Revenue*: In the case of profit-based provisioning, in order to maximize service profit, content items with higher potential revenue may need to be given higher priority. Thus items should have close revenue levels to be clustered together.
- *Delivery resource requirements*: When there are differences in the resource requirements of different content items, it is preferable to place items with high processing power requirements where processing cost is low and items with high bandwidth requirements where bandwidth cost is low. Therefore, items with similar requirement profile should be clustered together.
- *QoS requirements*: Content items with different quality of service requirements, such as last mile server-user distance restrictions represented by constraints (4.5c), (5.6c) or (6.8b), cannot be grouped into a cluster unless a more restrictive constraint is used for the cluster as a whole. However, we assumed that such items belong to different service classes and should be dealt with separately, and thus this attribute was not considered in developing clustering metrics.

In the next section, we will demonstrate that in failing to consider these content attributes, the existing clustering metrics, while performing well in some scenarios, would be inefficient in situations where key content attributes other than demand distribution are present.

7.3.2 Deficiencies in Existing Clustering Metrics

In applying existing content clustering techniques into ODN provisioning scenarios, we concentrated on the two correlation metrics that achieved the best performance in (Chen et al., 2003), namely clustering based on spatial demand distribution and clustering based on popularity.

- *Spatial demand distribution*: Given a set of N_a access nodes, the demand pattern for each content item k is described by a vector λ_{mk} , $m = 1, 2, \dots, N_a$ with N_a elements. Each item is then represented as a point in an N_a -dimensional space using this demand vector and the clustering distance between two items is the Euclidean distance between two points in this space¹.
- *Popularity*: In this simpler approach, the clustering distance between two items is defined as the difference in their access frequencies (i.e. total demand levels from all access nodes): $dist(k_1, k_2) = |\sum_m \lambda_{mk_1} - \sum_m \lambda_{mk_2}|$

We first applied these clustering metrics to the ODN provisioning for web content (optimization model (P) defined in (4.4)) in a network of 30 nodes and 1000 web pages. In this model the differences in content resource requirements are not considered and demand patterns remain the only factor that characterizes different content items, which is similar to situations considered in (Chen et al., 2003). It was found that while clustering content items based on spatial demand distribution is efficient as claimed in previous work, clustering based on popularity does not always give good performance and is highly dependent on demand distribution characteristics.

As an illustration, Figure 7.1 shows the results obtained by these clustering metrics when content requests contain a mixture of different demand patterns. These include globally, regionally and locally popular items, which are requested by 70 – 100%, 30 – 50% and 10% of access nodes, respectively. In this figure we plot ODN optimization results against the number of clusters, which is represented as a percentage of total number of individual content items. The optimization results (i.e. ODN

¹In the study by (Chen et al., 2003), the authors define the distance between two content items as their *correlation metric*. However, we believe this notation is not suitable and will use the term *clustering distance* in its place throughout this chapter.

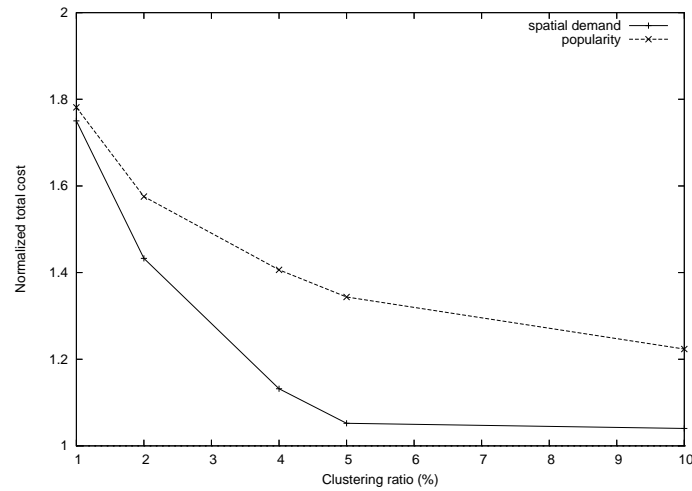


Figure 7.1 Clustering results for different clustering metrics in the ODN provisioning problem for web content with random demand patterns (lower is better)

topology costs) shown have been normalized by the results obtained by performing a full optimization without clustering. Thus in this graph, a lower plot indicates a better clustering metric, and the closer a plot is to 1, the better performing the metric is.

As can be seen in this figure, clustering by spatial demand distribution is indeed efficient. It is able to achieve solutions with quality comparable with a full optimization even when the number of clusters is as low as 5% of total number of content items, which is similar to results shown in (Chen et al., 2003).

Clustering based on popularity, on the other hand, is shown to have significantly lower performance. While seemingly different from existing results, this is actually not surprising. As it considers only the total access frequency, popularity-based clustering would only perform well when content items exhibit similar spatial demand distribution patterns. In the experiments by (Chen et al., 2003), access patterns were obtained from the most popular set of URLs in web proxy traces. Considering that web access generally have “long-tail” distribution patterns, where a large portion of accesses are due to a small portion of hot content items (Breslau et al., 1999; Erpanos et al., 2000), it is likely that many of the content items considered in the study were globally popular. This makes popularity the most important metric that captures the

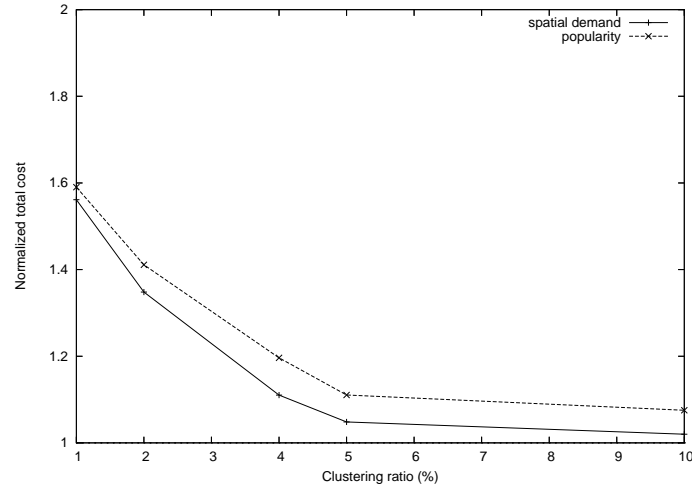


Figure 7.2 Clustering results for different clustering metrics in the ODN provisioning problem for web content with global demand patterns (lower is better)

access patterns. This is also noted by the authors.

In order to further confirm this suggestion, our previous experiment is repeated with a content population consisting of only globally popular items, which are accessed by at least 70% of access nodes. The results are then plotted in Figure 7.2, which again shows good performance by spatial clustering. As expected, the graph also shows markedly better performance by popularity-based clustering.

Content clustering based on spatial demand distribution, however, is not without drawbacks. Although it is seen to perform well in this web provisioning problem, it suffers low performance when applied to other more complex provisioning scenarios, such as the provisioning models defined in (5.5) and (6.7). This is because apart from demand distribution, these ODN provisioning problems also consider other important attributes to characterize content items, including revenue, processing and bandwidth requirements, which are not taken into account by the existing clustering scheme.

In order to illustrate the impact of differing content resource requirements and revenues on the performance of spatial demand clustering when used in provisioning models (5.5) and (6.7), we compared the results of several problems based on the same network topology but with different variation in content attributes.

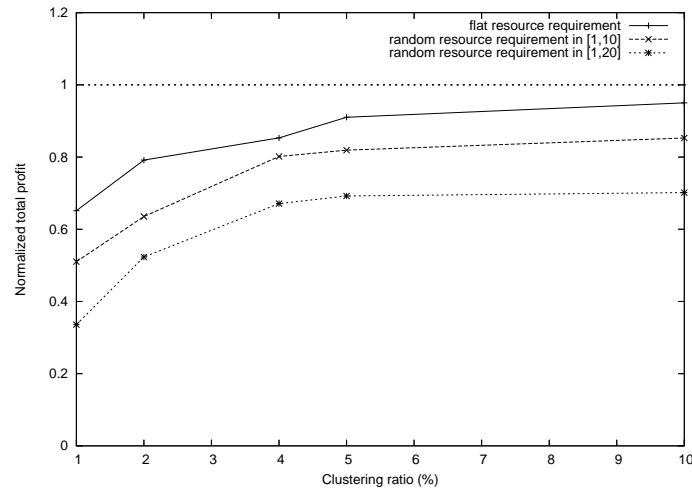


Figure 7.3 Results for spatial demand clustering in the ODN provisioning problem for pay-per-view content with different resource requirement variations (higher is better)

Figures 7.3 and 7.4 show the results obtained by the pay-per-view ODN provisioning model (5.5) for problems with 40 nodes and 1000 content items. The results shown have also been normalized by the output of a full optimization to enable comparing clustering performance in different problems. As this pay-per-view ODN provisioning model aims to maximize total service profit, a higher curve in these figures would indicate a better performance. Figure 7.3 compares the performance of spatial demand clustering with different content resource requirement variation, including: (1) same bandwidth and computation requirements for all items, (2) random bandwidth and computation requirements generated within range $[1, 10]$ and (3) random bandwidth and computation requirements generated within range $[1, 20]$. As can be seen from this figure, the performance of spatial demand clustering is dramatically reduced as resource requirement variation among content items increases. Figure 7.4, on the other hand, shows results obtained with increasing content revenue variation, including flat revenue, where revenue levels are the same for all items, and randomly generated revenues. This graph again shows a similar trend, with significant degradation in the performance of spatial demand clustering as content revenue variation increases. This is because the overlooked content revenue attribute is a key factor in assessing whether a content item should be replicated at all in this profit-based

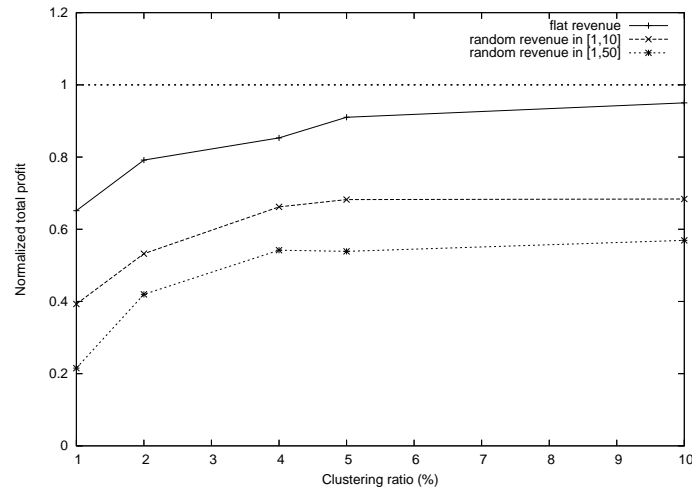


Figure 7.4 Results for spatial demand clustering in the ODN provisioning problem for pay-per-view content with different revenue variations (higher is better)

provisioning model.

In applying spatial demand clustering into the live streaming ODN provisioning model (6.7), a similar tendency is also observed, where the performance of spatial demand clustering is reduced when used in the presence of content resource requirement variation. This is illustrated in Figure 7.5, where a lower curve in this graph means better performance since the model aims to minimize the total ODN topology cost. As the graph clearly shows, the performance of spatial demand clustering is again negatively affected by increases in resource requirement variation amongst different content channels.

Results in this section have demonstrated that the existing content clustering metrics do not perform well in some ODN provisioning scenarios, as they fail to take into account a number of key factors that characterize content items in these problems. Clustering based on content popularity, for example, would not work well when content items do not have similar spatial demand patterns. This makes it unsuitable for use in ODN provisioning as in these problems not only simple web documents but also other types of content including pay-per-view and live streaming content are considered, for which typical demand distributions are practically uncharted territory and thus no assumptions could be made. Clustering based on spatial distribution,

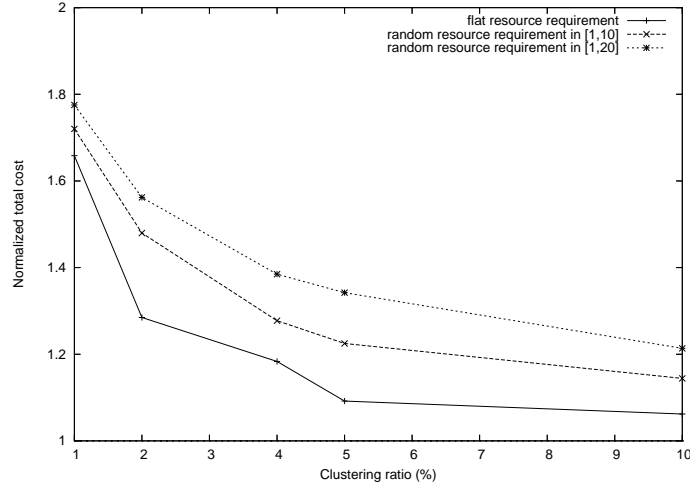


Figure 7.5 Results for spatial demand clustering in the ODN provisioning problem for live streaming content with different resource requirement variations (lower is better)

while performs better than the popularity-based method, also becomes inefficient in the presence of content revenue or resource requirement variations.

7.3.3 Modifying the Clustering Process

In this section, we aim to modify the clustering process in order to overcome existing weaknesses and improve its efficiency when used with ODN provisioning. Based on the observations in the previous sections, the following changes were proposed and examined:

1. *Hierarchical clustering*: In order to avoid grouping content items with significantly different resource requirement profiles into the same clusters, we devised a two-layer hierarchical clustering approach.

In the first layer, content items are clustered based on their delivery resource requirements. Each item k is represented by a resource requirement vector $[\bar{b}\gamma_k, \bar{p}\beta_k]$, which is weighted by average unit bandwidth and computation costs \bar{b} and \bar{p} to reflect the relative importance of each type of resource. Content items are then clustered based on Euclidean distances in this 2-dimensional space into a certain number of clusters.

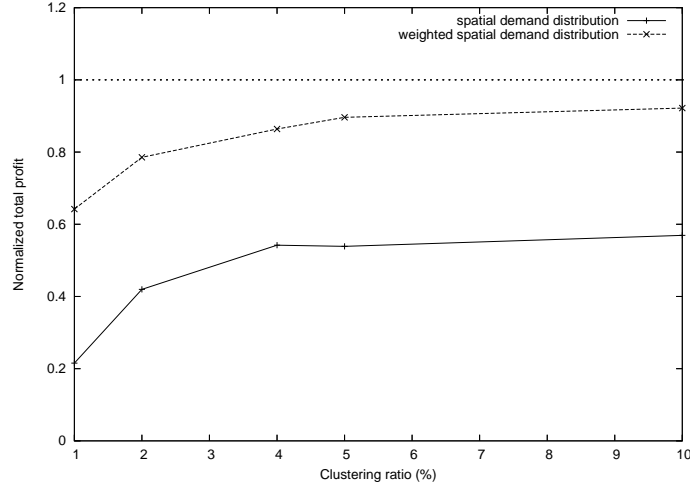


Figure 7.6 Comparing the performance of clustering by original spatial demand distribution and weighted demand distribution in ODN provisioning for pay-per-view content, where revenue is randomly generated between 1 and 50 (higher is better)

In the second clustering layer, items are further clustered using demand vectors without mixing items from different first-layer clusters. This is done by placing all items back in the N_a -dimensional space and repeatedly splitting the largest clusters into two, until a desired total number of clusters is reached.

2. *Weighting the demand vector*: when content revenue is considered by the provisioning process, the demand vector that describes the spatial demand distribution for an item is also weighted by its revenue level (i.e. $\rho_k \times [\lambda_{1k}, \lambda_{2k}, \dots, \lambda_{N_a k}]$). Weighting the demand vector with revenue levels effectively incorporates content priority into the clustering metrics, which is expected to help overcome the inefficiency of demand-distribution-only clustering in this situation. Note, however, that this is only required in the profit-based ODN provisioning problem (5.5), where content revenue factor is used.

7.3.3.1 Results

The effect of weighting demand vectors by content revenue on clustering performance is illustrated in Figure 7.6. This figure shows results obtained by both clus-

tering based on unweighted and weighted demand vectors, in the pay-per-view ODN provisioning problem with large content revenue variations previously used in Figure 7.4. As can be seen from the figure, weighting content demand vectors by their revenue levels has resulted in significant performance improvements in spatial clustering compared to the original unweighted demand vectors. This seems to be able to mitigate the weakness of spatial demand clustering in the presence of high revenue variation.

Figure 7.7, on the other hand, shows the results from the same provisioning model but for a problem with random variations in both content revenue and resource requirements. The problem has 1000 content items with revenue and resource requirements randomly generated in the range $[1, 10]$. The figure plots the normalized optimization output obtained with different clustering techniques, including (weighted and unweighted) spatial clustering and hierarchical clustering (with weighted demand vectors in the second clustering layer), against the final number of clusters. Results are also shown for hierarchical clustering with different number of clusters to be formed in the first clustering layer.

As seen from Figure 7.7, hierarchical clustering clearly produces significant improvements compared to non-hierarchical clustering, even when content items are just roughly classified (with number of first layer clusters smaller than 1% of total number of items). With hierarchical clustering, it is possible to achieve 90% solution quality at a number of final clusters as low as 5% of total number of items. We believe this is because even with a smaller number of first layer clusters, items are already effectively divided into groups that have similar resource emphases (e.g. heavy on bandwidth or processing).

When hierarchical clustering is used in the live streaming ODN provisioning model, results for problems with varying content resource requirements also show similar performance improvements over non-hierarchical clustering. For example, Figure 7.8 shows results from a problem with a total of 500 content channels and content resource requirements randomly generated from $[1, 20]$. This figure again confirms that hierarchical clustering has significantly better performance than non-hierarchical clustering when there are large resource requirement variations among different content channels. In this example, while the use of spatial demand clustering leads to

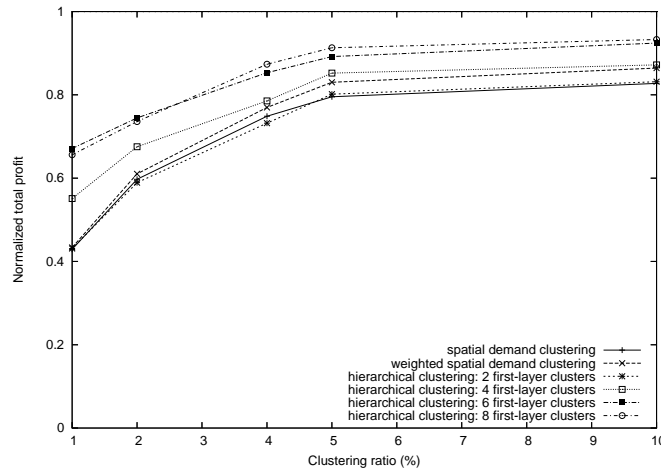


Figure 7.7 Hierarchical clustering in pay-per-view ODN provisioning problem (higher is better)

solutions significantly far (over 20%) from the full optimization result at a clustering ratio of 10%, hierarchical clustering was able to produce much better solutions (within 10% of full optimization result) even at a clustering ratio of 5%.

7.4 Conclusion

This chapter has introduced and examined the use of content clustering in ODN provisioning, whereby multiple similar content items are grouped together and treated as a single item, which helps improve the scalability of ODN provisioning models. We approached the problem by defining a general clustering framework that specifies how content clustering are incorporated into ODN provisioning processes, how content items are represented during provisioning by the clusters they belong to, and the objective of the clustering process.

A major part of the chapter was then devoted to finding a suitable content clustering scheme, in other words the criteria for content items to be grouped together. Although our study is inspired by and adopts the general concepts from an existing study on web content clustering for CDN replication by (Chen et al., 2003), we have shown that in ODN provisioning models the existing clustering metrics are inade-

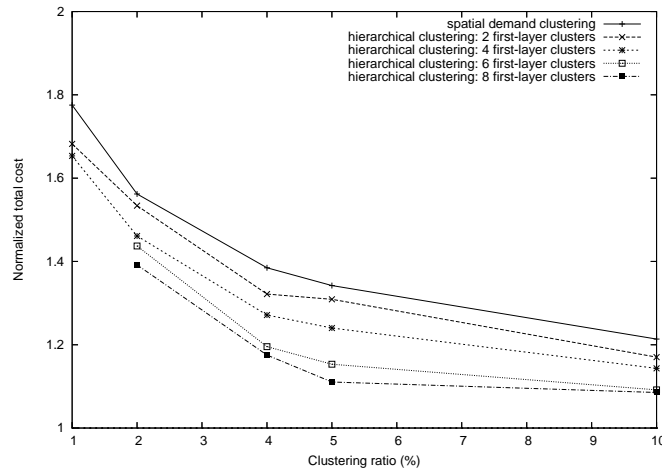


Figure 7.8 Hierarchical clustering in live streaming ODN provisioning problem (lower is better)

quate and need to be adapted to take into account new content characterizing factors. The existing study found that clustering web content pages based on either total access frequency (i.e. popularity) or spatial distribution of demand are the most effective approaches. We have demonstrated that the efficiency of clustering based on access frequency is highly dependent on the similarity of spatial demand distribution of different content items. This cannot be reasonably assumed for ODN applications under consideration, such as Internet-based pay-per-view or live streaming multimedia distribution. Clustering based on spatial distribution of demand, on the other hand, still works well when used with the provisioning model for web content ODN, but suffers significant performance degradation when used in other more complex ODN provisioning models. This is because in these cases there are other factors, including content revenue, processing and bandwidth requirements, that are also important to the clustering process but not considered in the existing clustering metrics. In view of these problems, we proposed a hierarchical content clustering process, in which content items are first grouped based on their resource requirement profiles, and then clustered based on their spatial demand distribution. The spatial demand vectors are also be weighted by content revenue in order to incorporate content priority into the clustering metric.

Experimental results have shown that the proposed hierarchical clustering approach has significant performance advantage compared to the existing non-hierarchical method. In our experiments it was possible to achieve solutions only 10% different from full optimization results at a number of clusters as low as 5% of the number of individual content items. In contrast, the original spatial demand clustering could produce results over 20% worse than full optimization even with twice the number of clusters.

A shortcoming of work in this chapter, however, is the fact that we have not produced a method of finding the optimal number of clusters to be found in each clustering layer of the hierarchical clustering process, which are found empirically in these experiments. For a given target final number of clusters, there is a trade off between the number of clusters to be generated by each clustering layer. The optimal numbers would depend on the particular problem, on whether the variations are more significant in content resource requirement profiles or in their spatial demand patterns. This is thus a difficult issue and is an open subject for further studies.

Chapter 8

Networking Support in ODN

8.1 Introduction

This chapter aims to examine the networking support required for communication among ODN servers and proposes to enhance the functionality of the ODN backbone with switching capabilities at server sites. This would allow application providers to control the forwarding of traffic flows within their overlay topologies.

In this chapter we will first establish the motivation for introducing this capability in ODN by examining its potential uses and benefits through qualitative and quantitative studies. Although the need for switching support may not be readily apparent in the current generation of content distribution applications, we believe it would enable better support for future applications where best effort transmission is no longer sufficient and network paths with certain quality of service (QoS) guarantees are required among servers. Examples of such applications include the distribution of interactive or high definition video content.

The benefits of switching support in ODN, as will be demonstrated in this chapter, include the following. Firstly, with switching support application providers would be able to control how traffic flows are routed within the ODN backbone, thereby utilizing indirect overlay paths via intermediate servers instead of relying solely on directly provisioned links or Internet routed paths. This would allow more flexibility in path establishment, because paths with end-to-end QoS could be achieved by combining QoS links purchased from different providers. Secondly, application providers

would also be able to switch traffic flows between alternative paths, which allows the implementation of dynamic path selection based on application-specific criteria. Additionally, the ability to use overlay paths would also eliminate the need for fully meshed backbone links and potentially enable more efficient backbone bandwidth usage by multiplexing multiple flows onto each virtual link.

We will also examine the challenges of providing this switching support in ODN and study the available implementation options. As it is not possible for application providers to deploy their own switching equipment in a virtual infrastructure environment, we envisage the introduction of shared switching resources, which could be leased along side network and server resources in order to perform switching operations in ODNs. We propose to achieve this by augmenting the shared infrastructure environment with a special type of shared switching device to be deployed at server sites. From these devices application providers will be able to lease switching resources in the form of virtual network switches, which belong to their own ODNs and perform traffic switching within these overlay topologies. Finally, we will describe an example shared switch architecture using overlay label switching technologies, which we believe is suitable for this purpose.

The chapter is organized as follows. Section 8.2 will present a qualitative study on the potential benefits of switching support in ODN and the reasons why it is required to enable better support for future applications. This will be further illustrated in Section 8.3, which aims to quantify the impacts of switching support on ODN bandwidth efficiency by developing and studying a number of ODN link provisioning models with different levels of switching support. Section 8.4 will then consider the possible options for providing switching support in ODN and propose, as the most suitable approach, the introduction of shared switching resources. A suitable shared switch architecture will also be identified and described in this section. Finally, concluding remarks will close the chapter in Section 8.5.

8.2 Why Switching Support?

The scenario in Figure 8.1 shows an example ODN topology with a number of virtual servers established in different locations. Depending on the application being

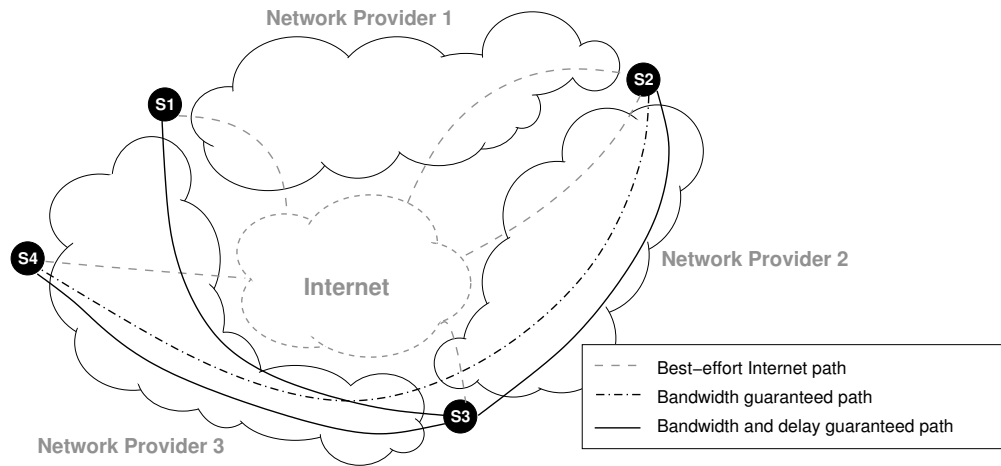


Figure 8.1 Example transport options for ODN backbone connectivity

delivered through this ODN, inter-server network connectivity may be required for a variety of reasons, including content distribution and update, back-end server communication or supporting other monitoring, control and management applications. Depending on the particular application's requirements and service availability, inter-server traffic could be supported with a range of different transport options, including best-effort Internet routed paths or QoS VPN connections provisioned directly between servers, as illustrated in Figure 8.1. The connections leased from different network providers could also vary in both QoS grades and underlying technologies. For example in this scenario QoS connections from Network Provider 2 may be created using MPLS Label Switched Paths, those from Network Provider 3 may be created using L2TP encapsulation tunnels, while Network Provider 1 provides no QoS service at all.

For applications that require no more than best-effort transmission quality to support its inter-server traffic, such as today's web content distribution, network connectivity provided by Internet paths is likely to be sufficient. However, for more demanding applications that place stricter quality of service requirements on these paths, for example high definition video distribution, virtual links with QoS guarantees may have to be leased alongside normal Internet connections to support traffic flows among servers. It is in such cases that switching support would become critical due to the

following reasons:

1. *Utilizing Overlay Connectivity:* Due to the large geographical span of ODN, network connectivity among servers will most likely be provided by a collection of different network providers, whose available services cannot be expected to be consistent in both quality and cost. Thus it would not always be possible or economical to lease virtual links with the adequate quality of service (e.g. bandwidth, delay or reliability guarantees) directly between every necessary server pair.

With control over traffic forwarding at each server site, however, application providers would be able to send traffic through alternative overlay paths instead of relying only on direct virtual links or Internet routed paths. This capability is extremely useful as it would allow alternative transmission paths to be established via non-direct links and give ODN providers more options and flexibility in selecting transport service providers and better chances in achieving the required QoS levels.

In the ODN scenario in Figure 8.1, for example, a direct QoS link cannot be leased directly between servers S_1 and S_4 . Without switching support, this may mean that high quality video content could not be delivered with to S_4 with sufficient play-ability. With switching support, however, an alternative overlay QoS path could be established via S_3 , which combine non-direct links using switching capabilities at the intermediate server.

2. *Resource Efficiency:* Even if there is no difficulty with virtual link availability or quality, the ability to use overlay paths would still be critical for efficient resource usage in the ODN backbone. Firstly, using overlay paths eliminates the need to provision a direct QoS link between every necessary server pair and the potential problem of establishing and managing an expensive fully meshed backbone. Secondly, sending traffic on overlay paths would enable each link to be shared by multiple traffic flows between different source and destination servers. This would allow more efficient usage of leased bandwidth through statistical multiplexing and lead to lower bandwidth requirements on inter-server links, as bandwidth requirement on each individual flow is likely

to fluctuate over time.

3. *Overlay Functionalities:* Switching support would also facilitate the implementation of overlay networking functionalities that are useful but may not be natively supported by the underlying networks, for example application-specific routing or overlay multicasting.

Application-specific routing could be used to support applications where distribution paths among servers are selected based on application-dependent QoS requirements and constraints rather than conventional routing metrics. For example an interactive application may require minimum-delay paths while a video streaming application may opt for paths with maximum bandwidth and low delay jitter. This would help applications to maintain their QoS requirements and react effectively to network condition changes. Such functionalities could be achieved by deploying overlay routing processes that determine the optimal path within the ODN topology and utilizing switching capabilities at each server site to ensure that traffic follows the desired paths.

Overlay multicast has been proposed in research literature as a viable alternative to network multicast and a solution to provide consistent multicast capability for content distribution over the Internet (Zhu et al., 2003; Andreev et al., 2003). Overlay multicast essentially combines packet duplication and hop-by-hop unicast transmission. It would be specially useful in the distribution of bandwidth-intensive content such as audio or video and could be implemented efficiently in the ODN if packet duplication and switching control are supported.

8.3 Switching and Bandwidth Efficiency

In this section we aim to further study and quantify the effects of switching support and overlay path forwarding on the efficiency of ODN backbone bandwidth provisioning.

We predict that by utilizing overlay paths and multiplexing multiple flows on each virtual link, bandwidth efficiency in the ODN backbone could be improved in the presence of bandwidth demand fluctuation in each individual flow over time. The

intuition is similar to the concept of *hose* bandwidth provisioning model in VPN literature, where bandwidth requirement for a multi-point VPN is significantly reduced by provisioning bandwidth at each VPN site as an aggregated amount to be shared by multiple paths and multiplexing traffic between different end points inside the network, instead of provisioning dedicated individual point-to-point bandwidth pipes (Duffield et al., 2002; Kumar et al., 2002). In this *hose* VPN model the underlying network infrastructure belongs to the same carrier, who would manage and route traffic flows among VPN end points in a way to maximize the bandwidth multiplexing. This could not be applied directly in infrastructure environment of ODN as the underlying network infrastructures belong to many different providers. However, we believe that similar multiplexing effects could still be achieved by leasing point to point connections among servers and then taking advantage of switching support to multiplex traffic flows over these links.

The effects of multiplexing on bandwidth efficiency will be quantified in this section by developing ODN backbone link provisioning models and comparing the resulting bandwidth costs in cases with and without switching support.

8.3.1 Network, Cost and Bandwidth Demand Models

We model the link provisioning problem as a graph $G = \langle V, E \rangle$, where nodes $i \in V$ represent locations of the established ODN servers and edges $(i, j) \in E$ represent potential QoS virtual links. Network connectivity among the servers could be provided by leasing either these dedicated, bandwidth guaranteed virtual links or best-effort Internet connections, or both.

Leasing a QoS virtual link from server i to server j is assumed to incur a link initiation cost of c_{ij} and unit bandwidth cost of b_{ij} . Best effort Internet connection bandwidth, on the other hand, is provisioned not on a point-to-point basis but as an aggregated gateway at each server, with unit bandwidth cost denoted g_i at server i . Bandwidth capacity at each server location i is also enforced by a maximum outgoing bandwidth (U_i) and incoming bandwidth (D_i).

In modelling inter-server traffic, we do not make any assumption on the nature of the application being delivered, but assume that the traffic flows among the servers could be estimated based on known factors such as server location, content loca-

tion, service and content characteristics, and user demand. We consider two types of flows: (1) Best effort flows that only require best effort Internet paths, and (2) QoS flows that must be sent through bandwidth guaranteed paths via specially provisioned QoS links, e.g. in the distribution of multimedia content. As the benefits of flow multiplexing would certainly depend on how much bandwidth requirement of each traffic flow varies over time, we aim to examine the multiplexing gains achieved with different degrees of demand fluctuation. The inter-server traffic pattern is therefore modelled as a multi-period, e.g. multi-hour or multi-day, traffic profile where the bandwidth requirement between each server pair may vary from one period to the next and the degree of demand fluctuation is then measured by the variation between different periods. We denote bandwidth demand for flows from server i to server j during time period t as ξ_{ij}^t for QoS traffic and λ_{ij}^t for best effort traffic.

For convenient reference, these parameters are summarized in the following list:

- c_{ij}, b_{ij} : QoS link initiation and unit bandwidth costs
- g_i : Best effort unit bandwidth cost
- U_i, D_i : Outgoing and incoming bandwidth capacities
- $\xi_{ij}^t, \lambda_{ij}^t$: Bandwidth demand matrices for QoS and best effort traffic

8.3.2 Link Provisioning Models

Given the set of server locations and traffic patterns among these servers, the aim of ODN link provisioning would be to find the optimal set of network links to be leased. This set of QoS and best effort Internet links should be able to support the given traffic flow matrices at all time periods with the minimum link and bandwidth cost.

A number of different provisioning models will now be developed that consider different levels of inter-server traffic switching, including: (1) application flow switching, where forwarding decisions are made per application flow, which is assumed to be significantly smaller than server-to-server flows. This allows each server-to-server flow to be spread over multiple overlay paths with different bandwidth splitting ratios

and is expected to give the best bandwidth multiplexing (2) server flow switching, where forwarding decisions are made per server-to-server flow, which allows only single path routing for each server flow, (3) no switching support, in which case no overlay paths are used and traffic is routed through either Internet connection or directly provisioned QoS virtual link. This is used as a comparison benchmark in order to evaluate the impact of switching support on ODN bandwidth efficiency.

8.3.2.1 Link Provisioning with Application Flow Switching

With application flow switching, multiple paths could be used to support each server-to-server traffic flow. QoS traffic may be sent through a combination of direct QoS links or overlay paths made up of multiple QoS links. Best effort traffic, on the other hand, could be sent through either best-effort Internet connections or existing QoS paths with spare capacities.

To formulate the provisioning problem in this case, the following variables are defined:

- $l_{ij} \in \{0, 1\}$: QoS link usage indicator, $l_{ij} = \begin{cases} 1 & \text{if link } (i, j) \text{ is leased} \\ 0 & \text{otherwise} \end{cases}$
- $x_{ij}^{sd,t}$: amount of QoS traffic from server s to server d being sent via link $(i, j) \in V$ in time period t .
- $y_{ij}^{sd,t}, z_i^{sd,t}$: amounts of best effort traffic from server s to server d being sent through QoS link (i, j) and through the Internet via intermediate server i , respectively
- u_{ij} : amount of QoS bandwidth leased on virtual link (i, j)
- v_i : amount of best effort Internet bandwidth leased at server i

with l_{ij} being binary variables and $x_{ij}^{sd,t}, y_{ij}^{sd,t}, z_i^{sd,t}, u_{ij}, v_i \in [0, \infty]$ being continuous variables. Note that continuous variables are used to measure traffic flows because it has been assumed traffic among servers could be spread over multiple paths with any splitting ratios.

With these variables and the input parameters defined above, the provisioning problem can now be formulated as follows:

$$(W_1) \quad \text{Min} \quad \sum_{(ij) \in E} l_{ij} c_{ij} + \sum_{(ij) \in E} u_{ij} b_{ij} + \sum_{i \in V} v_i g_i \quad (8.1)$$

Subject to:

$$\sum_{j|(i,j) \in E} x_{ij}^{sd,t} - \sum_{j|(j,i) \in E} x_{ji}^{sd,t} = \xi_{sd}^t, \quad \text{if } i = s, \forall s, d, t \quad (8.2a)$$

$$\sum_{j|(i,j) \in E} x_{ij}^{sd,t} - \sum_{j|(j,i) \in E} x_{ji}^{sd,t} = -\xi_{sd}^t, \quad \text{if } i = d, \forall s, d, t \quad (8.2b)$$

$$\sum_{j|(i,j) \in E} x_{ij}^{sd,t} - \sum_{j|(j,i) \in E} x_{ji}^{sd,t} = 0, \quad \text{if } i \neq s, i \neq d, \forall i, s, d, t \quad (8.2c)$$

$$\sum_{j|(j,i) \in E} y_{ji}^{sd,t} - \left(\sum_{j|(i,j) \in E} y_{ij}^{sd,t} + z_i^{sd,t} \right) = -\lambda_{sd}^t, \quad \text{if } i = s, \forall s, d, t \quad (8.3a)$$

$$\sum_{j|(j,i) \in E} y_{ji}^{sd,t} - \left(\sum_{j|(i,j) \in E} y_{ij}^{sd,t} + z_i^{sd,t} \right) = 0, \quad \text{if } i \neq s, i \neq d, \forall i, s, d, t \quad (8.3b)$$

$$\sum_{j|(j,i) \in E} y_{ji}^{sd,t} + \sum_{j \in V} z_j^{sd,t} = \lambda_{sd}^t, \quad \text{if } i = d \quad (8.3c)$$

$$\sum_{s \in V} \sum_{d \in V} x_{ij}^{sd,t} + \sum_{s \in V} \sum_{d \in V} y_{ij}^{sd,t} \leq u_{ij}, \quad \forall (ij) \in E, t \quad (8.4a)$$

$$\sum_{s \in V} \sum_{d \in V} z_i^{sd,t} \leq v_i, \quad \forall i, t \quad (8.4b)$$

$$u_{ij} \leq l_{ij} \times \min \{U_i, D_i, U_j, D_j\} \quad \forall i, j \quad (8.5a)$$

$$\sum_{j|(i,j) \in E} u_{ij} + v_i \leq U_i, \quad \forall i \quad (8.6a)$$

$$\sum_{j|(j,i) \in E} u_{ji} + \sum_{s \in V} \sum_{j \in V} z_j^{si,t} \leq D_i, \quad \forall i, t \quad (8.6b)$$

In this model, the objective function aims to minimize the total cost, including bandwidth guaranteed link initiation cost ($\sum_{(ij) \in E} l_{ij} c_{ij}$), their bandwidth cost ($\sum_{(ij) \in E} l_{ij} c_{ij}$),

and Internet bandwidth cost ($\sum_{i \in V} v_i g_i$). The constraints, on the other hand, are used to ensure that the leased backbone links are able to support the given traffic profile. Constraints (8.2a), (8.2b) and (8.2c) maintains flow conservation for QoS traffic from server s to server d , at the source, destination and intermediate nodes, respectively. Similarly, constraints (8.3a), (8.3b) and (8.3c) maintain flow conservation for best effort traffic flows from server s to d , during time period t . Constraints (8.4a) and (8.4b) ensures that the actual bandwidth usage at all times is within the amount provisioned, for QoS links and Internet connections, respectively. Constraint (8.5a) means bandwidth can only be provisioned on a virtual link if that link is leased. Finally, constraints (8.6a) and (8.6b) enforce bandwidth limits at each server location.

8.3.2.2 Link Provisioning with Server Flow Switching

The link provisioning model in this section is developed with the assumption that forwarding decisions in the ODN backbone are made per server-to-server flow, which means traffic flows of the same type, source and destination servers are to follow the same path, instead of splitting over multiple paths as in the previous model. This is considered in order to study the amount of bandwidth multiplexing that could still be achieved with coarser levels of flow manipulation.

As single-path routing is used, traffic flows will be described by the following new binary variables, which are defined to replace the continuous variables used in the previous model:

- $\bar{x}_{ij}^{sd,t}$: indicates if QoS traffic from server s to server d is being sent via link $(i, j) \in V$ in time period t
- $\bar{y}_{ij}^{sd,t}, \bar{z}_i^{sd,t}$: indicates if best effort traffic from server s to server d is being sent through QoS link (i, j) or through the Internet via intermediate server i , respectively, during time period t

With these new variables, the link provisioning model becomes:

$$(W_2) \quad \text{Min} \quad \sum_{(ij) \in E} l_{ij} c_{ij} + \sum_{(ij) \in E} u_{ij} b_{ij} + \sum_{i \in V} v_i g_i \quad (8.7)$$

Subject to:

$$\sum_{j|(i,j) \in E} \bar{x}_{ij}^{sd,t} - \sum_{j|(j,i) \in E} \bar{x}_{ji}^{sd,t} = 1, \text{ if } i = s, \forall s, d, t \quad (8.8a)$$

$$\sum_{j|(i,j) \in E} \bar{x}_{ij}^{sd,t} - \sum_{j|(j,i) \in E} \bar{x}_{ji}^{sd,t} = -1, \text{ if } i = d, \forall s, d, t \quad (8.8b)$$

$$\sum_{j|(i,j) \in E} \bar{x}_{ij}^{sd,t} - \sum_{j|(j,i) \in E} \bar{x}_{ji}^{sd,t} = 0, \text{ if } i \neq s, i \neq d, \forall i, s, d, t \quad (8.8c)$$

$$\sum_{j|(j,i) \in E} \bar{y}_{ji}^{sd,t} - \left(\sum_{j|(i,j) \in E} \bar{y}_{ij}^{sd,t} + \bar{z}_i^{sd,t} \right) = -1, \text{ if } i = s, \forall s, d, t \quad (8.9a)$$

$$\sum_{j|(j,i) \in E} \bar{y}_{ji}^{sd,t} - \left(\sum_{j|(i,j) \in E} \bar{y}_{ij}^{sd,t} + \bar{z}_i^{sd,t} \right) = 0, \text{ if } i \neq s, i \neq d, \forall i, s, d, t \quad (8.9b)$$

$$\sum_{j|(j,i) \in E} \bar{y}_{ji}^{sd,t} + \sum_{j \in V} \bar{z}_j^{sd,t} = 1, \text{ if } i = d \quad (8.9c)$$

$$\sum_{s \in V} \sum_{d \in V} \bar{x}_{ij}^{sd,t} \xi_{sd}^t + \sum_{s \in V} \sum_{d \in V} \bar{y}_{ij}^{sd,t} \lambda_{sd}^t \leq u_{ij}, \forall (ij) \in E, t \quad (8.10a)$$

$$\sum_{s \in V} \sum_{d \in V} \bar{z}_i^{sd,t} \lambda_{sd}^t \leq v_i, \forall i, t \quad (8.10b)$$

$$u_{ij} \leq l_{ij} \times \min \{U_i, D_i, U_j, D_j\} \quad \forall i, j \quad (8.11a)$$

$$\sum_{j|(i,j) \in E} u_{ij} + v_i \leq U_i, \quad \forall i \quad (8.12a)$$

$$\sum_{j|(j,i) \in E} u_{ji} + \sum_{s \in V} \sum_{j \in V} \bar{z}_j^{si,t} \lambda_{si}^t \leq D_i, \quad \forall i, t \quad (8.12b)$$

The structure of this problem formulation is very similar to the previous one (W_1), except that traffic flows are now described by binary variables and cannot be split over multiple forwarding paths. Subsequently, the constraints have also been adjusted to accommodate this change.

8.3.2.3 Link Provisioning without Switching

When switching support is lacking, overlay forwarding paths cannot be used and inter server flows must travel via best-effort Internet connections or direct virtual

links. Thus no flow multiplexing will occur and the required links and bandwidth can be calculated in a more straightforward manner, to be used as a benchmark in assessing bandwidth efficiency.

Assuming that it is possible to lease direct QoS connections between all necessary server pairs, the bandwidth to be leased on virtual link from server i to server j to support QoS traffic is simply the maximum amount required in all time periods:

$$\hat{u}_{ij} = \max_t \{\xi_{ij}^t\} \quad (8.13)$$

Assuming that QoS bandwidth is more expensive than best effort bandwidth, best effort inter-server traffic most of the time would be forwarded normally via the Internet connections, and would be sent via QoS links only if there is enough spare capacity, which is already leased but unused in some periods. Therefore best effort bandwidth requirements for flow from server i to server j in period t would be:

$$\hat{v}_{ij}^t = \begin{cases} 0 & \text{if } \hat{u}_{ij} - \xi_{ij}^t \geq \lambda_{ij}^t \\ \lambda_{ij}^t & \text{otherwise} \end{cases} \quad (8.14)$$

The amount of Internet bandwidth to be leased at server i thus could be calculated as follows:

$$\hat{v}_i = \max_t \left\{ \sum_j \hat{v}_{ij}^t \right\} \quad (8.15)$$

As a result, the total provisioning cost for ODN backbone links and bandwidth can be found by:

$$(\mathbf{W}_3) \text{ Total cost} = \sum_{(ij) \in E} \hat{l}_{ij} c_{ij} + \sum_{(ij) \in E} \hat{u}_{ij} b_{ij} + \sum_{i \in V} \hat{v}_i g_i \quad (8.16)$$

where $\hat{l}_{ij} = \begin{cases} 1 & \text{if } \hat{u}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$ is the virtual link usage indicator.

8.3.3 Numerical Results

In order to compare the inter-server bandwidth usage with different levels of switching support considered, we implemented provisioning models (W_1) , (W_2) and (W_3) using optimization package Cplex (Cplex, 2001) and studied the numerical results, which were obtained with input problems based on network topologies generated by

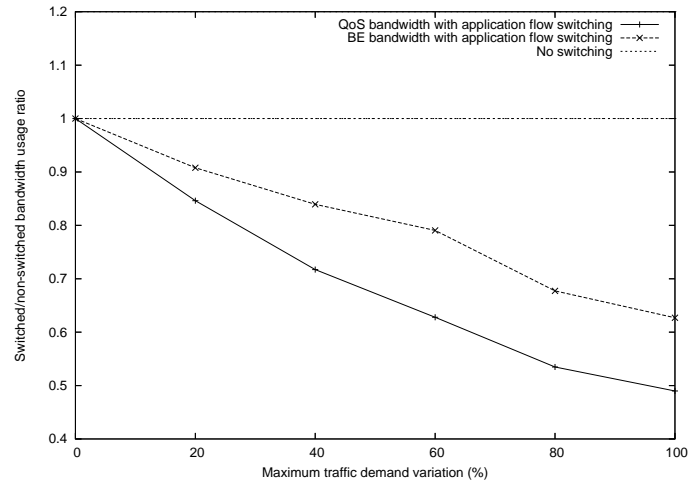


Figure 8.2 Bandwidth cost with application flow switching

the GT-ITM topology generator.

In these generated topologies, the nodes indicate ODN server locations and edges indicate potential QoS virtual links. We assumed bandwidth costs to be proportional to the distance between servers (i.e. edge length) and initiation costs to be zero for all these links, which makes bandwidth cost the only component in provisioning objectives. This allows us to put aside the obvious full-mesh link cost disadvantage of the scenario without switching support and compare bandwidth efficiency of the different cases on a more equal ground.

Inter-server traffic patterns, on the other hand, are modelled by a series of flow matrices for different time periods, with each matrix element specifying the bandwidth requirement between a certain pair of servers. This is created by starting with set of flows as the average demand, and introducing fluctuations into each flow to create demand levels for individual time periods. As we aim to study the effects of flow multiplexing at different levels of flow bandwidth variation, the fluctuations are controlled and measured by the maximum deviation from the average values. For example, if the initial bandwidth requirement of a flow is λ , with an $x\%$ deviation the value for individual time periods would be generated randomly between $\max\{0, \lambda(1 - x/100)\}$ and $\lambda(1 + x/100)$. The same procedure is used to create bandwidth demand for both QoS and best effort traffic flows.

Numerical results from all experiments consistently showed that the use of overlay

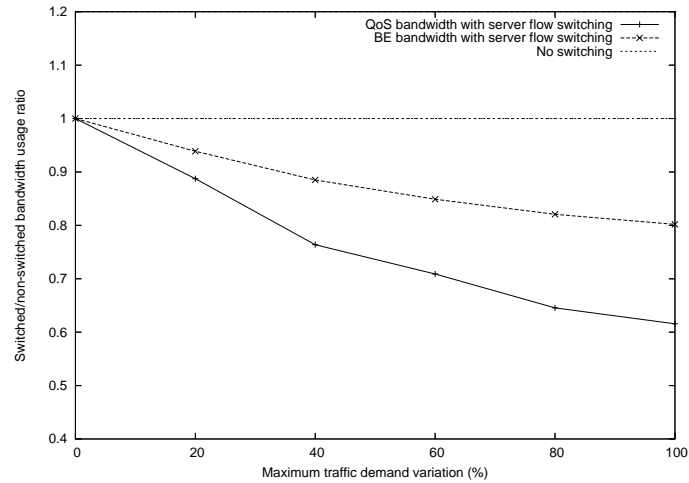


Figure 8.3 Bandwidth cost with server flow switching

paths and application flow switching leads to significant reductions in bandwidth requirement, with total bandwidth cost appears to decrease almost linearly with traffic fluctuation levels.

This can be seen from Figure 8.2, which shows the results from a problem with 20 servers, over 300 flows for each traffic class and a multi-period traffic profile consisting of 10 time periods. The figure plots QoS and best-effort bandwidth costs achieved with application flow switching for a range of maximum flow bandwidth deviation between 0 and 100%. When plotting the graphs in this figure, bandwidth costs with switching are normalized by the corresponding costs without switching, which allows bandwidth reduction ratios due to flow multiplexing to be observed. The horizontal line at $y = 1$ thus represents the costs required if switching is not supported. As the figure shows, bandwidth requirements on backbone links can be reduced dramatically with switching if there are significant traffic fluctuations in individual flows. For example when the maximum traffic variation is 100%, the total bandwidth required on QoS links are 50% of what would be required without switching support.

When server flow switching is used and the forwarding of each flow is restricted to a single path, numerical results from provisioning model (W_2) indicated that the effectiveness of flow multiplexing on link bandwidth is reduced, as expected. However, even in this case bandwidth reduction due to switching is still significant, especially

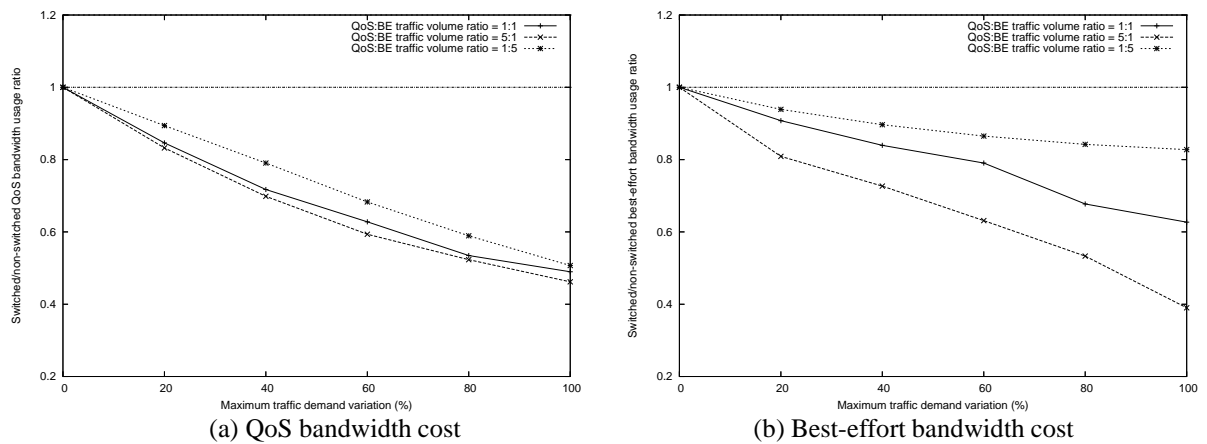


Figure 8.4 Switched/non-switched bandwidth cost ratio achieved with different Best-effort:QoS traffic volume ratios

on QoS virtual links. This can be seen from Figure 8.3, which shows bandwidth costs achieved with server flow switching for the same traffic requirements used in Figure 8.2, which shows bandwidth savings of up to 40% on QoS links under very high traffic fluctuation.

Comparing the multiplexing effects on best-effort and QoS bandwidth, it was also observed that while multiplexing gains on the dedicated virtual links remain consistent with the trends in the previous figures, best-effort Internet bandwidth gains are variable and very dependent on the relative volumes of best-effort and QoS traffic. This is demonstrated by Figure 8.4, which show results from problems with average QoS:best-effort flow bandwidth requirement ratios of 1:1, 1:5 and 5:1. As can be seen from Figure 8.4(a), which plots the normalized QoS bandwidth usage for these cases, the multiplexing gains on QoS links are close among the three problems and are very similar to the trend observed earlier in Figure 8.2. Best-effort bandwidth gains, however, varies from one problem to another and appears to be more significant with the presence of higher QoS traffic volume. This is actually logical and can be explained by the fact that the main benefit of switching and overlay forwarding on best-effort bandwidth usage is the ability to divert best-effort traffic flows via QoS paths in order to take advantage of unused spare capacities on already leased links and save on Internet bandwidth. This, however, would become more likely and more

significant as greater amounts of QoS bandwidth are leased.

8.3.4 Remarks

The numerical results above clearly showed that with switching support, inter-server flows could be manipulated and multiplexed effectively on overlay paths in order to reduce bandwidth requirements in the ODN backbone, especially when bandwidth requirements of server-to-server traffic flows fluctuate significantly over time. Although not illustrated in the experiments, it is also reasonable to expect that cost savings due to switching support in the ODN backbone would become even more significant if virtual link initiation costs are high, as with the use of overlay paths less links would be required.

A limitation in the provisioning models developed above is the fact that we have considered only bandwidth guarantees but not other factors such as delay, jitter or reliability requirements on QoS traffic flows. However, we believe these models have been beneficial in providing further insight into the effects of switching on ODN bandwidth efficiency and numerical proof of its benefits.

8.4 Providing Switching Support in a Shared Infrastructure Environment

Given that switching support is desirable and useful in enhancing the functionality, flexibility and resource efficiency of the ODN, this section aims to examine the challenges in providing such capabilities and to propose the concept of shared switching resources as a solution.

8.4.1 The Challenge

In the traditional CDN model, where server sites are deployed, owned and operated by the application provider, providing overlay networking functionalities would be reasonably straightforward, as standard or application specific networking equipment such as routers, switches or specialized middle boxes could be deployed at each site to form an overlay network on top of the underlying carrier networks. This is an ap-

proach that could probably be employed by existing CDN owners in order to enable better support for next generation applications.

In a shared infrastructure environment where the overlay network is created over shared infrastructure by leasing virtual servers and network links, however, application providers have no control over the physical infrastructure and thus would not be able to deploy their own networking equipment.

A simple option would be to implement the required functionalities in software to be processed inside virtual servers. The servers would then double as packet processing and switching nodes that relay traffic between different virtual links. In this approach, application providers have the freedom to implement any extra networking functionalities required by their applications on top of the standard protocol stacks and no special cooperation would be required among the different server providers. However, on the downside, performing packet processing using servers' general-purpose CPUs would be costly and inefficient, which adds extra burden on the servers. This could also introduce significant delay and unpredictable jitter due to varying server processing load, which may seriously affect the quality of overlay network paths.

8.4.2 The Proposed Solution

In order to overcome these problems, we propose to augment the shared infrastructure environment with shared switching resources by deploying specially designed network switches that are capable of identifying, intercepting and performing network operations on selected traffic flows in hardware and could be used by multiple application providers simultaneously. We envisage that such switches would be deployed by server providers between server sites and their network providers, and offer switching resources to application providers in the form of "virtual switches". Each "virtual switch" would serve as an end point for a number of virtual links in an application provider's overlay topology and allow rules to be defined for selected traffic flows to be processed and switched among these links. By leasing such virtual switches together with server resources at server sites, an application provider effectively obtains an overlay infrastructure of servers, network connectivity and consistent networking support without burdening the servers with these functions.

This approach would require the integration of shared switches into the existing

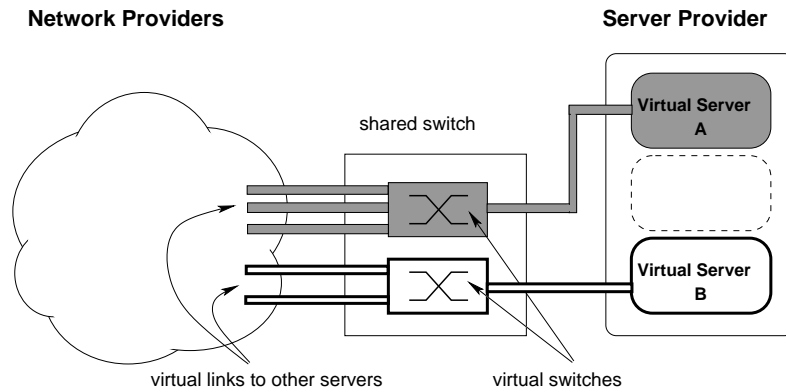


Figure 8.5 Shared switching resource concept

server infrastructure as well as a certain degree of cooperation among different server providers so that virtual switches leased at different locations could interoperate. However, the benefits of switching support as presented so far in this chapter applies equally not only to content distribution but also to any other distributed-server applications deployed in the shared infrastructure environment. Thus we believe that there are incentives for server owners to deploy these switches and that the resulting enhancement would be critical to the adoption and eventual success of the shared infrastructure paradigm.

The shared switching resource concept is illustrated by the diagram in Figure 8.5, which shows virtual links, virtual switches and virtual servers established at the same server site by two different application providers. As the diagram indicates, traffic flows belonging to each application provider would be managed and processed according to the rules defined in that provider's virtual switch. Furthermore, due to the switch's special location, overlay traffic could be intercepted and forwarded on without going through intermediate servers.

A major challenge in designing this shared switching device is to provide efficient traffic processing capabilities to multiple application providers while restricting operations by each provider to his own traffic flows. In the next section we will describe an example switch architecture that we believe could be used to fulfill this requirement.

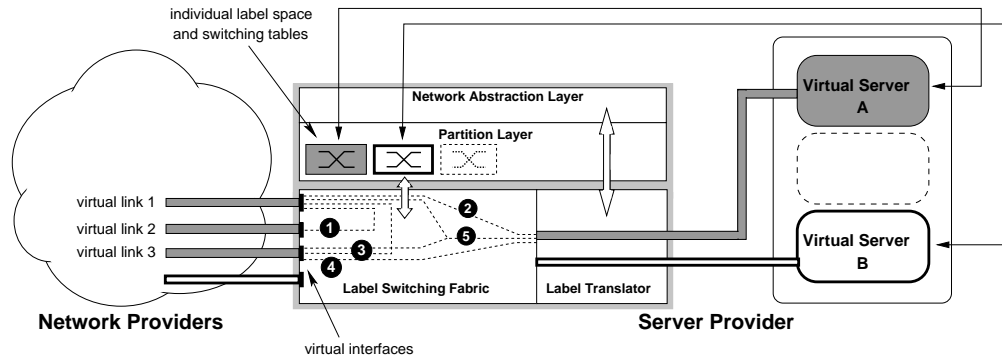


Figure 8.6 A Shared Label Switch Architecture

8.4.3 A Shared Switch Architecture

The shared switch architecture that we propose is illustrated Figure 8.6. This architecture is based on opening the control plane of network switches and allowing application providers to define processing rules for their own traffic flows. Although the original concept was developed in a larger project that this thesis project is a part of, substantial modifications and implementation details have been introduced in order to make this architecture suitable for the ODN environment.

The major building block of this architecture is a network switch capable of processing and forwarding packets based on labels instead of IP addresses, using label switching technologies such as Multi Protocol Label Switching (MPLS) (Rosen et al., 2001) or Tag Switching (Rekhter et al., 1997). With these technologies, packets carry an additional field called a *label* or *tag*, usually in a “shim” header inserted between network and datalink layer headers. Packets may also carry more than one labels in a structure called a label stack. How packets should be handled, including forwarding, swapping, adding or removing labels, is specified in a switching table based on the top label and the path that a packet follows is called a Label Switched Path (LSP).

Through virtual links leased by application providers, neighboring relationships would be established among these switches. These links would appear from the switches’ point of view as direct connections and would be presented to the switches’ control plane as virtual interfaces, through which labelled traffic flows could be forwarded.

To achieve this, the switch must also be able to perform the necessary processing required to send packets through these connections, which may be different at different sites depending on the underlying physical network infrastructure and the specific tunneling mechanisms used by network providers to create virtual links. This could be as simple as adding additional labels when virtual links are provided in the form of LSPs but could also involve complex encapsulation when virtual links are provided as encapsulation tunnels.

By creating overlay connections among switches, purchasing a set of labels in each switch, defining switching rules based on these labels and labeling traffic flows appropriately, application providers would be able to control and manipulate traffic flows among their servers. As illustrated by numbered paths in Figure 8.6, switching operations to be supported in the switch would include:

1. Switching between different virtual links. This could be used to join multiple virtual links and create overlay label switched paths that traverse multiple links but bypass intermediate servers.
2. Switching between a virtual link and a virtual server.
3. Switching between a virtual link and an Internet routed path.
4. Switching between a virtual server and an Internet routed path.
5. Packet duplication, which could be used in conjunction with switching to implement overlay multicast.

In order to allow application providers to define switching rules while restricting such control to their own traffic flows, the control plane of the switch is augmented with a “Partitioning” layer that keeps track of the virtual interfaces and label space created by each application provider and maintains a separate switching table for each. Application providers can access and modify the content of their switching tables. However, only switching rules among the same application provider’s interfaces can be created. The operating system of the switch would then be responsible for translating and merging the individual tables into the switch’s main switching table.

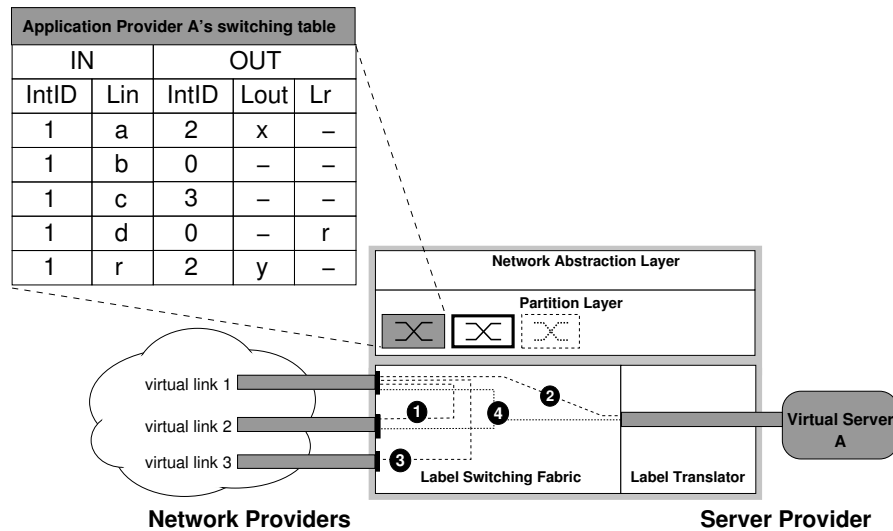


Figure 8.7 Example switching operations

To further enforce the separation between application providers, access to the individual switching tables are made through an interface called the “Network Abstraction” layer. This layer hides, i.e. abstracts, the actual virtual interface and label values assigned to each application provider’s LSPs in order to prevent misuse and traffic impersonation. The interface also ensures that application providers only access switching tables that they own and gives each application provider the impression of a private “virtual switch” that belongs to the application provider’s overlay topology. Because of this abstraction, labelled packets passing between virtual servers and the switch will be intercepted by a “Label Translator”, which swaps between abstracted label values used by application providers and real label values used inside the network. As labels have a fixed size and are stored at a fixed location in packet headers, this translation could be done easily and efficiently.

8.4.3.1 Example Switching Operations

The operation of the switch is illustrated in Figure 8.7 with a number of packet switching examples.

Figure 8.7(b) shows a switch, a virtual server and a set of virtual links that belong to application provider A. Note that from the switch’s perspective, each of these links is

associated with a virtual interface, through which labeled packets could be sent and received. The individual switching table for provider A is also shown in Figure 8.7. Each entry in this table corresponds to a switching rule and contains the incoming virtual interface ($IntID$), incoming packet label, (Lin), outgoing interface, outgoing label ($Lout$), and a special field called *recycled label* (L_r), which is used for packet duplication purposes. To illustrate the switching process, we consider packets arriving through virtual link 1 with the following different labels:

- *Label a*: Packets carrying this label will match the first entry of the table. Thus these packets will be forwarded on through virtual link 2 with the label changed to x . This switching plus label swapping operation is the commonly used standard capability in label-switching network switches.
- *Label b*: Packets carrying this label will match the second rule. Assuming that 0 designates the switch interface connected to the servers, these packets will be forwarded on towards the virtual server.
- *Label c*: Packets labeled with c will be sent on through virtual link 3, which is actually a best effort Internet connection. The outgoing label field is empty in this case as the packets would be stripped of their labels, forwarded on towards their destinations using default Internet routed paths and no encapsulation is required.
- *Label d*: Packets carrying label d will match the fourth rule and will be forwarded towards the virtual server in a similar fashion to those with label b above. However, since the recycled application label is non empty, while each packet is being processed, an identical copy will be made, but with a new label r , the value in the L_r field. This packet is then processed based on the new label and get forwarded on through virtual link 2 with application label y . This packet duplication is intended to be used in implementing overlay multicast functionality.

The advantage of using label switching in this architecture is the fact that switching rules are defined based on packet labels, while the classification and labelling of

packets rests with application providers. This gives significant flexibility in traffic flow control and manipulation as inter-server traffic could be classified and switched in different granularities depending on application requirements. For example, all traffic from one server to another could be labelled with a single label or split into multiple flows with multiple labels depending on the packets' application and/or QoS requirements.

8.4.3.2 Implementation Directions

Although the implementation of this shared switch architecture is out of the scope of this thesis and is left for future work, we have explored a number of possible implementation approaches.

Ideally, the switch should be implemented with all functional building blocks integrated and packet processing carried out in dedicated specialized hardware. This approach, however, requires significant hardware knowledge, design effort and investment and would perhaps only be pursued in the manufacturing of final products ready for mass deployment.

For a proof-of-concept prototype, an implementation could also be created in software, for example using Linux-based MPLS label switching capabilities provided in the *Linux MPLS* implementation by (NetX, 2001). Such an implementation would allow further study and evaluation of the switch architecture and its functionalities, and allow any unforeseen requirements or difficulties to be discovered and addressed. This, however, would still require significant effort as every functional building block of the switch has to be emulated, and performance characteristics would also be hard to assess in a software-based implementation.

In order to take advantage of existing label switching hardware, it would also be possible to implement the shared switch architecture by augmenting existing standard label switches (e.g. MPLS-capable network routers and switches) with new functionalities provided in the form of middle boxes or line cards that are plugged into the existing switch's interfaces. This approach is illustrated in Figure 8.8, where functional blocks such as the Partition layer, Network Abstraction layer and Label Translator are implemented in an add-on middle box situated between the server farm and the switch. This component is used to control the configuration of the main

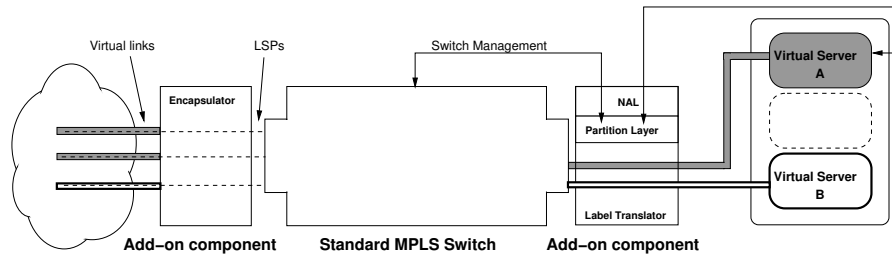


Figure 8.8 Implementing a shared switch by extending an existing MPLS switch

switch, possibly by communicating with the switch through a management interface. It could also be used to perform operations that are not normally supported in standard switches, such as packet duplication used in overlay multicasting.

If some form of encapsulation is required to send packets through virtual links, an encapsulation stage may need to be deployed in front of the switch as another line card or middle box, which also serves as the terminator for network tunnels. As existing standard label switches may not support the creation of virtual interfaces associated with application providers' individual virtual connections, a work-around could be used where strictly different label spaces are allocated for different virtual links, and the encapsulator would then look at packet labels to determine the correct virtual link to forward packets on. This could also be further simplified by having an extra label indicating the outgoing virtual link pushed onto packets before forwarding, an operation that existing label switches are capable of. The encapsulator could then simply look at this top label to determine the correct outgoing virtual link. These complex operations, however, could be made transparent to the application providers, who will only see virtual switches with interfaces associated with their virtual links, label spaces and switching tables that they could control.

The advantage of this last approach is the fact that only the extra functionalities have to be implemented and existing resources including label switching hardware, software and related protocols, e.g. signalling protocols for LSP creation among switches, could be utilized.

Compared to standard label switching operations, we have introduced a number of extra processing steps into the data path, including label translation and possibly en-

capsulation. However these components have very simple and specific functions and thus could be implemented in application-specific hardware to improve performance and minimize effects on transmission path quality.

8.5 Conclusion

In this chapter we have considered the ability of ODN topologies to support future applications where inter-server transmission paths with quality of service guarantees are required. We argued that in such situations relying solely on directly provisioned virtual links or Internet routed paths for inter-server connectivity may not be sufficient. We proposed to enhance the virtual overlay network topologies with switching support that allows application providers to control the forwarding of traffic within their overlay networks, thereby manipulating traffic flows and utilizing overlay paths via intermediate servers. The following major issues have then been examined and addressed in this chapter: Firstly, the benefits of switching in supporting applications with QoS path requirements using overlay network topologies have been demonstrated. We have shown that this capability is critical in meeting inter-server path QoS requirements, because it allows alternative overlay paths to be used when direct virtual links with adequate end-to-end QoS cannot be leased. It is also required to allow application providers to deploy overlay networking functionalities such as application-specific path selection or overlay multicast.

In addition, switching is expected to eliminate the need for an expensive full mesh of inter-server QoS links and allow better bandwidth efficiency through multiplexing flows on overlay paths. In order to confirm this, we have developed bandwidth provisioning models for multi-period inter-server traffic demand profiles in scenarios with and without switching support. Two levels of flow manipulation were also considered: application flow switching, where each server-to-server flow could be split into small application flows and switched over multiple overlay paths, and server flow switching where each server-to-server flow uses only a single path. Numerical results obtained with these models showed that switching allows significant reductions on virtual link bandwidth cost when bandwidth requirements of individual server-to-server flows fluctuate over time. For example when the bandwidth peak of each

individual flow is up to twice its average value, total bandwidth costs on virtual links could be reduced by about 50% with application flow switching and 40% with server flow switching.

Secondly, we have pointed out that providing switching support for overlay network topologies in a shared infrastructure environment is a significant challenge. Deploying private switching hardware is infeasible not only because application providers have no control over physical infrastructure but also because switching would be required by multiple application whose overlay topologies co-exist over the same physical infrastructure. In light of this challenge, we proposed to provide switching support by deploying shared switching resources, which could be leased by application providers in the form of “virtual switches” that belong to their own overlay topologies.

For this purpose, we have developed and described a suitable shared switch architecture to be deployed at server sites by server owners. The architecture is based on existing label-switching network switches and allows application providers to lease virtual switches and define switching rules to manipulate their traffic flows in different granularities. A number of possible implementation directions for this architecture have also been identified for use in future work.

Chapter 9

Conclusions

9.1 Overview

Emerged within the last decade, Content Distribution Networks (CDN) have quickly been recognized as a useful method for improving Internet content distribution scalability and quality. By replicating content to a world-wide network of servers and redirecting users to nearby servers based on server proximity, server load, content availability or network conditions, a CDN could help improve content access latency, avoid server overloading and bypass network congestion. However, a drawback of the current CDN model is the high deployment cost. This is because currently deploying a CDN requires a private server infrastructure spanning the Internet, which is prohibitively expensive.

In search of an alternative, more cost effective and more flexible content network deployment approach, in this thesis we proposed the Overlay Distribution Network (ODN) concept. Instead of deploying dedicated infrastructure, we propose to lease transport and server resources from server and network providers to establish an overlay network of virtual servers and connecting virtual links, which is then used to deliver content in a similar manner to today's CDNs. This approach is expected to require less resource commitment from the application providers and also creates the possibility of adjusting the content network's topologies or capacities on-the-fly according to demand. Considering the current trend of resource virtualization and sharing and infrastructure outsourcing in the Internet, we believe this is quite within

the grasp of today's technologies.

The ODN approach, however, would also significantly transform the way content networks are provisioned and content replicas are placed. We have shown that the shared infrastructure nature of this approach would introduce new requirements into the ODN provisioning process compared to traditional CDNs, including new objectives, new constraints and the fact that resource provisioning and content replication have to be addressed jointly. As these new requirements are not yet satisfied by existing works in current literature, we went on to address them by formulating and studying ODN provisioning models for a number of content applications that we believe would be important in the future Internet. These include traditional web content distribution, pay-per-view content distribution and live streaming multimedia content distribution.

In order to further improve the computation scalability of the provisioning models to handle problems with very large number of content items, we then explored the incorporation of content clustering into the ODN provisioning process. By grouping similar content items together into clusters, which are then considered as a single item by the provisioning process, content clustering could be used to reduce the provisioning complexity.

Finally, we turned our attention to the inter-server communication in the ODN backbone and proposed to further enhance the ODN architecture with switching support that allows ODN owners to control the flow of traffic within their ODN backbone links. We examined the benefits of such capabilities and proposed a possible shared switch architecture that could be used to provide such support in a shared infrastructure environment.

The following section of this chapter will present the major conclusions from each of these studies.

9.2 Major Conclusions

9.2.1 ODN Provisioning Framework

By reviewing related works in CDN research literature and analyzing the characteristics of a shared infrastructure environment, we have pinpointed the following major differences between provisioning problems in ODNs and traditional CDNs:

- Compared to CDN provisioning, the ODN provisioning problem is set in a more complex environment with more factors to consider. For example there could be a large number of potential infrastructure providers, whose available resources could be offered for different prices, with different quality of service grades and in different quantities. Such factors affect the desirability of leasing virtual servers and virtual links at each particular location and need to be considered by the provisioning process in order to make cost-effective provisioning decisions.
- ODN provisioning involves topology planning, resource dimensioning and replica placement problems, which are interdependent and have to be addressed jointly. In contrast, CDN replica placement is often addressed in current literature as a stand-alone problem, which looks for the optimal content replication pattern given an established CDN topology.
- ODN provisioning would also involve a very different objective. In an ODN, leasing more resources may lead to better performance but would also incur higher resource cost. Thus the objective of ODN provisioning would be to find the optimal topology that delivers satisfactory application performance at the minimum resource cost, instead of searching for the best performance possible as commonly seen in CDN literature.

Based on these observations, an ODN framework was developed, which describes how provisioning would be carried out in ODNs. According to this framework, ODN provisioning would involve an application-specific optimization that determines the optimal ODN topology, content replication and request routing patterns. This process

requires information from both infrastructure providers and the application provider, and is expected to be carried out by the application provider. This framework has served as a guideline in the development of application-specific ODN provisioning models throughout this thesis.

9.2.2 Provisioning ODN for Web Content Distribution

In Chapter 4 we studied the provisioning of web content ODNs and formulated a provisioning model that not only captures the characteristics of this problem but also avoids over-simplifications that are seen in many of the existing CDN replica placement models. We also developed and studied a number of heuristics, including a Lagrangian relaxation heuristic, a two-level greedy search heuristic and a simple greedy search heuristic, in order to find near optimal solutions for this NP-hard problem.

From this study we have found that efficient heuristics are an invaluable tool in tackling the ODN provisioning problem. For example while solving the provisioning model for optimal solution is only practical in very small networks, the proposed Lagrangian and two-level greedy heuristics were able to find near optimal solutions for problems that are orders of magnitude larger in size.

We demonstrated that the performance of different heuristics also varies greatly. For example, the Lagrangian relaxation and two-level greedy search heuristics were able to find ODN topologies that cost within 36% and 51% of optimal, respectively. The CDN-based greedy heuristic, on the other hand, performed significantly worse and produced topologies up to 2.5 times more costly. This is because it was not designed based on the structure of the problem formulation as the other two heuristics.

We also found that while the Lagrangian heuristic generally produces better results, it requires significantly more computation time. Thus this heuristic would be a good choice when the provisioning problem size is not too large. For larger problems where computation time becomes an issue, the greedy heuristic would be a better candidate.

9.2.3 Provisioning ODN for Pay-Per-View Content Distribution

In Chapter 5 we studied the ODN provisioning problem in a pay-per-view content distribution scenario. We focused on the class of applications that distribute static content such as recorded TV programs or movies through the Internet in a pay-per-view manner. We considered a distribution model where the ODN owner also owns the content and is able to decide whether a given movie or recorded show should be replicated based on its potential profit.

To address this problem, we formulated an ODN provisioning model that aims to maximize the service profit. In this model, not all content items have to be replicated. Rather, available items are assessed and selected for replication based on their potential profit as part of the provisioning process. We have then demonstrated that for the class of pay-per-view content distribution applications in question, this provisioning model is the most suitable. Alternative provisioning models that have no or stand-alone item selection methods were shown to perform significantly worse, resulting in up to 5 times less total profit.

We have also developed an efficient Lagrangian heuristic that exploits the formulated provisioning model's structure to break it down into a series of very small and simple subproblems. Experimental results showed that the heuristic is able to find near optimal solutions, which are within 20% from optimal in all experiments, with impressive scalability. For example solutions close to optimal for problems of up to 120 nodes and 1000 content items could be found within hours while looking for the exact solution is practical only for very small problems of under 20 nodes and 20 content items.

9.2.4 Provisioning ODN for Live Multimedia Content Distribution

In Chapter 6 we studied the distribution of live streaming multimedia content through ODNs. In this scenario ODN servers would function as both streaming proxies and processing nodes where content enhancement functions are deployed. We also propose to implement overlay multicast within the ODN backbone to facilitate the distribution of live content from sources to servers.

We showed that with this type of application, the ODN provisioning problem would be significantly different from those previously considered. For example, the provisioning process would need to establish not only front end streaming servers but also backbone servers, backbone links and content distribution trees. Furthermore, the possibility of using a particular server location not only depends on its resources and cost but also whether a distribution path with adequate quality could be established from the content source.

To address this challenge we formulated an ODN provisioning model that determines not only the optimal ODN server and link configuration, but also content distribution trees in the ODN backbone. An efficient Lagrangian heuristic was then developed to find near optimal solutions to this NP-hard problem. The heuristic was able to achieve reasonable solution accuracy (within 20% of optimal) and could solve problems with 120 nodes and 500 content channels in the order of hours.

We then studied the use of a separate online optimization process to fine-tune the content distribution process in order to accommodate inaccurate demand estimation or short term demand fluctuations, given an established ODN topology. This process is intended to be used over short time scales, during which re-provisioning already allocated resources is not practical. This online optimization could involve redirecting/rebalancing content requests or reorganizing content distribution trees. A number of alternative online optimization schemes were developed and evaluated, including a *full optimization* that could reorganize distribution trees as well as re-balance demand among servers, a *partial optimization* that only re-balances demand, and a *local optimization*, which is a simple distributed optimization to be carried out independently at each server to selectively admit content requests based on a utility value if the server is under-provisioned.

We found that with small demand fluctuations (up to 20%), the local optimization method is adequate, while with greater fluctuations partial optimization can maintain good performance in most cases. Only in the presence of abrupt geographical demand shifts is a full optimization required. Overall, using online optimization allows service performance to be maintained above 80% of total utility for demand deviations up to 60%. This indicates that the provisioning and online optimization models together form an efficient provisioning tool, capable of creating topologies that could

handle large service demand fluctuations.

9.2.5 Improving ODN Provisioning Scalability with Content Clustering

In Chapter 7, we explored the incorporation of content clustering into our ODN provisioning models in order to improve provisioning scalability in terms of number of content items. By grouping content items into clusters, which are then treated as a single item by the provisioning process, provisioning complexity could be reduced drastically, although at possible loss of solution quality.

The major challenge was to find a suitable content clustering scheme that could significantly reduce the content item count without significantly compromising solution quality.

Although we adopted the clustering concept from an existing study on web content clustering for CDN replication by (Chen et al., 2003), we showed that in ODN provisioning models the existing clustering metrics are inadequate as they fail to take into account other significant factors that could affect clustering efficiency, including content delivery resource requirements and revenue.

In the existing study it was found that clustering web content pages based on either popularity or spatial distribution of demand are the most effective approaches. We have found that clustering based on access frequency would perform poorly if content items have different spatial demand distribution patterns. We also found that clustering based on spatial distribution of demand, while adequate for the ODN provisioning model for web content, suffers significant performance degradation when used in other more complex ODN provisioning models that take into account different content resource requirements and revenues among content items.

In view of these problems, we proposed a hierarchical content clustering process, in which content items are first grouped based on their resource requirement profiles, and then clustered based on their spatial demand distribution. The spatial demand vectors are also weighted by content revenue in order to incorporate content priority into the clustering metric.

Experimental results showed that the proposed hierarchical clustering scheme with weighted demand vectors has significant performance advantages compared to the

existing scheme. In our experiments it was possible to achieve solutions only 10% different from full optimization results with number of clusters as low as 5% of the number of individual content items. The original spatial demand clustering, in contrast, produce results over 20% worse than full optimization even with twice the number of clusters.

9.2.6 Networking Support in ODN

In Chapter 8 we examined the inter-server communication in the ODN backbone and proposed to enhance the ODN architecture with switching support that allows application providers to control the forwarding of inter-server traffic. Two major issues were addressed in this chapter:

Firstly, we demonstrated that switching support in ODN is beneficial, especially in the presence of applications that require QoS paths for inter-server traffic. This is because with switching support alternative overlay paths could be used instead of relying only on direct virtual links with adequate end-to-end QoS, which may not always be feasible. Switching would also allow application providers to deploy useful overlay networking functionalities such as application-specific path selection or overlay multicast.

In addition, we predicted that switching would eliminate the need for an expensive full mesh of inter-server QoS links and allow better bandwidth efficiency through multiplexing flows on overlay paths. We then confirmed this by developing and studying bandwidth provisioning models for multi-period inter-server traffic demand profiles in scenarios with different levels of switching support. It was found that switching support indeed allows significant reductions on virtual link bandwidth cost when bandwidth requirements of individual server-to-server flows fluctuate over time. For example when the bandwidth peak of each individual flow is up to twice its average value, total bandwidth costs on QoS virtual links could be reduced by 50% with application flow switching and 40% with server flow switching.

Secondly, we identified the challenges in providing switching support for ODNs and proposed a solution based on shared switching resources. This allows application providers to lease switching resources in the form of “virtual switches” that belong to their own overlay topologies. A suitable shared switch architecture based on label-

switching network switches was then described, which enables application providers to lease virtual switches and define switching rules to manipulate their traffic flows in different granularities. A number of possible implementation directions for this architecture have also been identified for use in future work.

9.3 Summary of Major Contributions

The major contributions of work in this thesis can be summarized as follows:

- Proposed the Overlay Distribution Network concept as an alternative, more cost effective and more flexible content network deployment strategy compared to traditional CDNs.
- Identified how the new deployment approach would transform the way content networks are provisioned and content replicas are placed. As a result an ODN provisioning framework was developed that specify how provisioning process would be carried out for ODNs.
- Developed a provisioning model and efficient heuristics for ODNs that distribute web content.
- Developed a provisioning model and efficient heuristic for ODNs that distribute pay-per-view content.
- Developed a provisioning model and efficient heuristic for ODNs that distribute live streaming multimedia content. A separate online optimization process that fine tunes the distribution process in order to better utilize already allocated resources in the presence of demand fluctuation was also developed and found to be effective.
- Improved the scalability of ODN provisioning models using content clustering. Deficiencies in existing content clustering methods were identified and a new scheme based on hierarchical clustering was proposed, which was found to significantly improve clustering performance.

- Demonstrated the benefits of introducing switching support in ODNs with both qualitative and quantitative evidence.
- Identified the challenges of providing switching support in ODN, proposed a suitable solution based on shared switching resources and identified a possible shared switch architecture.

9.4 Future Work

The possible directions for future research to improve or extend the work presented in this thesis include:

- Improving hierarchical content clustering: Experiments in this thesis have shown that a hierarchical content clustering approach performs significantly better when used in ODN provisioning, compared to other clustering methods. However, we have not produced a method of finding the optimal number of clusters to be created in each clustering layer of this hierarchical clustering process. For a given target final number of clusters, there is a trade off between the number of clusters to be generated by each clustering layer. This would depend on the parameters of the particular problem, for example whether the variations are more significant in content resource requirement profiles or in their spatial demand patterns. This is still an open subject for further research in the future.
- Improving provisioning scalability in terms of network size: The use of content clustering, as explored in Chapter 7 of this thesis, only helps improve ODN provisioning scalability in terms of content item count. Further research is needed in order to improve the scalability of ODN provisioning models in terms of network size. Such improvement would allow provisioning problems based on networks significantly larger than what have been considered in this thesis to be handled. A possible direction to achieve this could be the use of segmentation algorithms that effectively break down large networks into smaller segments. Each individual segment could then be handled by a separate

provisioning process. The challenge would be to find a segmentation method that does not lead to significant degradation in the overall provisioning quality.

- From the study of online optimization in ODNs for live streaming content distribution in Chapter 6, we have learned that the use of online optimization is extremely beneficial to the performance of the ODN when there are short-term demand fluctuations. We believe this would also apply to other ODN provisioning models developed for web content and pay-per-view content in this thesis. Therefore the work presented in this thesis could be further improved by developing new online optimization algorithms or adapting the developed ones for use in these models.
- The output of ODN provisioning process considered in this thesis include request routing parameters. However, we have not studied the integration of this output and request routing models used during the ODN's operation. The research issues that merit further investigations include: (1) How the request routing parameters resulting from the provisioning process should be communicated to and used by the request routing system, and (2) how the performance of the ODN would be affected when the actual request routing pattern drifts from the desired pattern due to the imperfections of the request routing system, for example because of the limitations of DNS-request routing as discussed in Chapter 2.

Bibliography

Accelia (2005). www.accelia.net.

Acharya, S. and Smith, B. (2000). Middleman: A Video Caching Proxy Server. In *Proc. NOSSDAV 2000*. IEEE.

Agarwal, G., Shah, R., and Walrand, J. (2001). Content Distribution Architecture Using Network Layer Anycast. In *IEEE WIAPP 2001*, pages 124–132.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, New Jersey.

Akamai (2005). www.akamai.com.

Alex, V. C. (1992). A Global File System. In *Proc. 1992 USENIX File System Workshop*, page 112.

Allen, A. (2001). Optimal Delivery of Multi-Media Content over Networks. In *Proc. the Ninth ACM International Conference on Multimedia*, Ottawa, Canada.

Almeida, J. M., Eager, D. L., Ferris, M., and Vernon, M. K. (2002). Provisioning Content Distribution Networks for Streaming Media. In *Proc. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2002*, volume 3, pages 1746–1755. IEEE.

Andreev, K., Maggs, B. M., Meyerson, A., and Sitaraman, R. K. (2003). Designing Overlay Multicast Networks for Streaming. In *Proc. the Ninth ACM International Conference on Multimedia*, pages 149 – 158, San Diego.

- Andrews, M., Shepherd, B., Srinivasan, A., Winkler, P., and Zane, F. (2002). Clustering and Server Selection using Passive Monitoring. In *Proc. IEEE INFOCOM 2002*.
- Aqun, Z., Yuan, Y., Yi, J., and Guanqun, G. (2000). Research on Tunneling Techniques in Virtual Private Networks. In *Proc. International Conference on Communication Technology, 2000*, pages 691 – 697, Beijing, China.
- Barbir, A., Cain, B., Nair, R., and Spatscheck, O. (2003). Request for Comments 3568: Known Content Network (CN) Request-Routing Mechanisms, IETF CDI Working Group.
- Barford, P. and Plonka, D. (2001). Characteristics of Network Traffic Flow Anomalies. In *ACM SIGCOMM Internet Measurement Workshop*.
- Barish, G. and Obraczke, K. (2000). World Wide Web caching: Trends and Techniques. *IEEE Communications Magazine*, 38(5):178–184.
- Biliris, A., Cranor, C., Douglass, F., Rabinovich, M., Sibala, S., Spatscheck, O., and Sturm, W. (2001). CDN Brokering. In *Sixth International Workshop on Web Caching and Content Distribution*, Boston, Massachusetts, USA.
- Biliris, A., Cranor, C., Douglass, F., Rabinovich, M., Sibala, S., Spatscheck, O., and Sturm, W. (2002). CDN Brokering. *Elsevier Computer Communications*, 25:393–402.
- Bradley, A. D. and Bestavros, A. (2002). Basis Token Consistency: Supporting Strong Web Cache Consistency. In *Global Internet 2002 (GI2002)*, Taipei, Taiwan.
- Breslau, L., Pei, C., Li, F., Phillips, G., and Shenker, S. (1999). Web Caching and Zipf-Like Distributions: Evidence and Implications. In *Proc. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'99*, pages 126–134, New York.
- Cahill, A. and Sreenan, C. (2003). VCDN: A Content Distribution Network for High-Quality Video. In *Proc. Information Technology & Telecommunications Conference (IT&T)*.

- Cahill, A. and Sreenan, C. (2005). An Efficient CDN Placement Algorithm for the Delivery of High-Quality TV Content. In *Proc. 9th IASTED International Conference on Internet and Multimedia Systems and Applications (EuroIMSA)*.
- Calvert, K., Doar, M., and Zegura, E. W. (1997). Modeling Internet Topology. *IEEE Communications Magazine*.
- Cao, P. and Liu, C. (1997). Maintaining Strong Cache Consistency in the World-Wide Web. In *Proc. Seventeenth International Conference on Distributed Computing Systems*.
- Chan, S.-H. G. and Tobagi, F. (2001). Distributed Servers Architecture for Networked Video Services. *IEEE/ACM Transactions on Networking*, 9(2):125–136.
- Chankhunthod, A., Danzig, P. B., Neerdaels, C., Schwartz, M. F., and Worrel, K. J. (1996). A Hierarchical Internet Object Cache. In *1996 USENIX Technical Conference*, San Diego, CA.
- Charikar, M., Guha, S., Tardos, ., and Shmoys, D. B. (2002). A Constant-Factor Approximation Algorithm for the K-Median Problem. *Journal of Computer and System Sciences*, 65(1):129–149.
- Chase, J. S. (2001). Server Switching: Yesterday and Tomorrow. In *The Second IEEE Workshop on Internet Applications, WIAPP*, pages 114–123.
- Chen, H., Zhao, W., and Xie, L. (2004). A DNS-Pertinent Routing Algorithm with the Maximum Network Revenue in Content Distribution Networks. In *Proc. the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, volume 1, pages 121–124.
- Chen, Y., Katz, R. H., and Kubiawicz, J. D. (2002). SCAN: A Dynamic, Scalable, and Efficient Content Distribution Network. In *International Conference on Pervasive Computing*.
- Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. (2003). Efficient and Adaptive Web Replication using Content Clustering. *IEEE Journal on Selected Areas in Communications*, 21(Y).

- Cidon, I., Kутten, S., and Soffer, R. (2001). Optimal Allocation of Electronic Content. In *Proc. INFOCOM 2001*, pages 1773–1780.
- Cleary, K. (1995). Video on Demand - Competing Technologies and Services. *International Broadcasting Convention*, (413):432–437.
- Cplex (2001). ILOG Cplex Optimization Suite: www.ilog.com/products/cplex.
- Cunningham, D. and O'Mahony, D. (1995). Secure Pay-Per-View Testbed. In *Proc. the International Conference on Multimedia Computing and Systems 1995*, pages 308–311. IEEE.
- Day, M., Cain, B., Storigen, Tomlinson, G., and Rzewski, P. (2003). Request for Comments 3568: A Model for Content Internetworking (CDI), ETF CDI Working Group.
- Dilleya, J. and Arlitt, M. (1999). Improving Proxy Cache Performance: Analysis of Three Replacement Policies. *IEEE Internet Computing*, 3(6):44–50.
- Domingo-Ferrer, J. and Martinez-Balleste, A. (2002). STREAMOBILE: Pay-Per-View Video Streaming to Mobile Devices over the Internet. In *Proc. 13th International Workshop on Database and Expert Systems Applications 2002*, pages 418 – 422.
- Duffield, N. G., Goyal, P., Greenberg, A., Mishra, P., Ramakrishnan, K. K., and van der Merwe, J. E. (2002). Resource Management With Hoses: Point-to-Cloud Services for Virtual Private Networks. *IEEE/ACM Transactions on Networking*, pages 679–692.
- Eagera, D. L., Vernon, M. K., and Zahorjan, J. (2000). Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand. In *Proc. 2000 Multimedia Computing and Networking*, San Jose, CA.
- EdgeStream (2005). www.edgestream.com.
- Erpanos, D. N., Karakostas, G., and Wolf, W. H. (2000). Effective Caching of Web Objects using Zipf's Law. In *Proc. IEEE International Conference on Multimedia and Expo, ICME2000*, pages 727–730, New York.

- F. K. Hwang, a. D. S. R. and Winter, P. (1992). *The Steiner tree problem*. Amsterdam ; New York : North-Holland.
- Fei, Z. (2001). A Novel Approach to Managing Consistency in Content Distribution Networks. In *WCW'01. IEEE International Web Content Caching and Distribution Workshop*.
- G., C. S. H. and Tobagi, F. (2001). Distributed Servers Architecture for Networked Video Services. *IEEE/ACM Transactions on Networking*, 9(2):125–136.
- Glover, F. and Laguna, M. (1993). *Modern Heuristic Techniques for Combinatorial Problems*, chapter Lagrangean Relaxation. Blackwell, Oxford, UK.
- Green, M., Cain, B., Tomlinson, G., Thomas, S., and Rzewski, P. (2002). Internet Draft: Content Internet Working Architectural Overview, IETF CDI Working Group.
- Gsieniec, L., Jansson, J., and Lingas, A. (2004). Approximation Algorithms for Hamming Clustering Problems. *Journal of Discrete Algorithms*, 2(2):289–301.
- GTL (2001). The Graph Template Library: infosun.fmi.uni-passau.de/GTL.
- Hama, T., Asatani, K., and Nakazato, H. (2004). P2P Live Streaming System with Low Signal Interruption. In *Proc. 18th International Conference on Advanced Information Networking and Applications, AINA 2004*, volume 1, pages 605 – 610.
- Hochbaum, D. S. and Shmoys, D. B. (1985). A Best Possible Approximation Algorithm for the K-Center Problem. *Mathematics of Operations Research*, 10:180–184.
- Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. (2000). On the Placement of Internet Instrumentation. In *Proc. INFOCOM 2000*, pages 295–304.
- Jamin, S., Jin, C., Kurc, A. R., Raz, D., and Shavitt, Y. (2001). Constrained Mirror Placement on the Internet. In *Proc. INFOCOM*, pages 31–40.

- Johnson, K. L., Carr, J. F., Day, M. S., and Kaashoe, M. F. (2000). The Measured Performance of Content Distribution Networks. In *The 5th International Web Caching and Content Delivery Workshop, WCW'00*, Lisbon, Portugal.
- Joo, H. (2003). Private and Fair Pay-Per-View Scheme for Web-Based Video-On-Demand Systems. *IEEE Transactions on Consumer Electronics*, 49(2):403–407.
- Jung, J., Krishnamurthy, B., and Rabinovich, M. (2002). Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *Proc. the International World Wide Web Conference*, pages 252–262. IEEE.
- Kangasharju, J., Roberts, J., and Ross, K. (2001a). Object Replication Strategies in Content Distribution Networks. In *Proc. WCW'01: Web Caching and Content Distribution Workshop*, Boston, MA.
- Kangasharju, J., Roberts, J., and Ross, K. (2002). Object Replication Strategies in Content Distribution Networks. *Elsevier Computer Communications*.
- Kangasharju, J., Ross, K. W., and Roberts, J. W. (2001b). Performance Evaluation of Redirection Schemes in Content Distribution Networks. *Computer Communications*, 24(2):207–214.
- Karlsson, M., Karamanolis, C., and Mahalingam, M. (2002). A Framework for Evaluating Replica Placement Algorithms. Technical report, Hewlett-Packard Laboratories.
- Karlsson, M. and Mahalingam, M. (2002). Do We Need Replica Placement Algorithms in Content Delivery Networks. In *WCW'01. IEEE International Web Content Caching and Distribution Workshop*.
- Katabi, D. and Wroclawski, J. (2000). A Framework for Scalable Global IP-Anycast (GIA). In *SIGCOMM*, pages 3–15.
- Korupolu, M., Plaxton, G., and Rajaraman, R. (2001). Placement Algorithms for Hierarchical Cooperative Caching. *Journal of Algorithms*, 38(1):260–302.

- Krishnan, P., Raz, D., and Shavitt, Y. (2000). The Cache Location Problem. *IEEE/ACM Transactions on Networking*.
- Kumar, A., Rastogi, R., Silberschatz, A., and Yener, B. (2002). Algorithms for Provisioning Virtual Private Networks in the Hose Model. *IEEE/ACM Transactions on Networking*, pages 565–578.
- Lee, N. (2000). Fairness and Privacy on Pay-Per View System for Web-Based Video Service. *IEEE Transactions on Consumer Electronics*, 46(4):980–985.
- Li, B., Golin, M. J., Italiano, G. F., Deng, X., and Sohraby, K. (1999). On the Optimal Placement of Web Proxies in the Internet. In *Proc. INFOCOM 1999*.
- Li, Y. and Liu, M. T. (2003). Optimization of Performance Gain in Content Distribution Networks with Server Replicas. In *Proc. the 2003 Symposium on Applications and the Internet (SAINT'03)*. IEEE.
- Martello, S. and Toth, P. (1990). *Knapsack Problems*. Wiley.
- Mirkin, B. G. (1996). *Mathematical Classification and Clustering*. Kluwer Academic Publishers, Dordrecht; Boston.
- Miura, H. and Yamamoto, M. (2002). Content Routing with Network Support using Passive Measurement in Content Distribution Networks. In *Proc. Eleventh International Conference on Computer Communications and Networks*, pages 96–101. IEEE.
- Mukhtar, R. and Rosberg, Z. (2003). A Client Side Measurement Scheme for Request Routing in Virtual Content Distribution Networks. In *IEEE International Performance, Computing, and Communications Conference*, Phoenix, AZ USA.
- Nemhauser, G. L. and Wolsey, L. A. (1999). *Integer and Combinatorial Optimization*. Wiley.
- NetX (2001). Linux MPLS Implementation:
www.cl.cam.ac.uk/Research/SRG/netos/netx.
- Nguyen, T. V., Chou, C., and Boustead, P. (2003a). Provisioning CDN over Shared Infrastructure. In *Proc. IEEE ICON 2003*, pages 119–124, Sydney.

- Nguyen, T. V., Chou, C., and Boustead, P. (2003b). Resource Optimization for CDN over Shared Infrastructure. In *Proc. Australian Telecommunications Networks and Applications Conference*, Melbourne.
- Nguyen, T. V., Safaei, F., and Boustead, P. (2004). Provisioning Overlay Distribution Networks for Live Streaming Media. In *Proc. Australian Telecommunications Networks and Applications Conference*, Sydney.
- Nguyen, T. V., Safaei, F., Boustead, P., and Chou, C. T. (2005). Provisioning Overlay Distribution Networks. *Computer Networks*, 49(1):103–118.
- Ninan, A., Kulkarni, P., Shenoy, P., Ramamritham, K., and Tewari, R. (2002). Cooperative Leases: Scalable Consistency Maintenance in Content Distribution Networks. In *11th World Wide Web Conference (WWW-2002)*, Honolulu, Hawaii.
- Norton, R. L. (2002). Using Virtual Linux servers. *Computer*, 35(11):106–107.
- Qiu, L., Padmanabham, V., and Voelker, G. (2001). On the Placement of Web Server Replicas. In *Proc. 20th IEEE INFOCOM*.
- Radoslavov, P., Govindan, R., and Estrin, D. (2001). Topology-Informed Internet Replica Placement. In *Proc. WCW'01: Web Caching and Content Distribution Workshop*, Boston, MA.
- Ramanathan, S. (1996). Multicast Tree Generation in Networks with Asymmetric Links. *IEEE/ACM Transactions on Networking*, 4(4).
- Rekhter, Y., Davie, B., Rosen, E., Swallow, G., Farinacci, D., and Katz, D. (1997). Tag Switching Architecture Overview. In *Proc. the IEEE*, volume 85, pages 1973–1983.
- Rosen, E., Viswanathan, A., and Callon, R. (2001). Request for Comments 3031: Multiprotocol Label Switching Architecture, IETF Network Working Group.
- Rzewski, P., Day, M., and Gilletti, D. (2003). Request for Comments 3570: Content Internetworking (CDI) Scenarios, IETF CDI Working Group.
- Sandbank, C. P. (2001). Digital TV in the Convergence Environment. *IEEE Computer Graphics and Applications*, pages 32–36.

- Sen, S., Towsley, D., Zhang, Z. L., and Dey, J. K. (2002). Optimal Multicast Smoothing of Streaming Video over the Internet. *IEEE Journal on Selected Areas in Communications*, 20(7):1345–1359.
- Shaikh, A., Tewari, R., and Agrawal, M. (2001). On the Effectiveness of DNS-based Server Selection. In *Proc. IEEE INFOCOM 2001*, Anchorage, AK.
- Shim, S. and Lee, Y.-J. (2002). Interactive TV: VoD meets the Internet. *IEEE Computer*, 35(7):108–109.
- Speedera (2005). www.speedera.com.
- Sridharan, R. (1995). The Capacitated Plant Location Problem. *European Journal of Operational Research*, 87:203–213.
- Srinivasan, R., Liang, C., and Ramamritham, K. (1998). Maintaining Temporal Coherency of Virtual Warehouse. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS98)*, Madrid, Spain.
- Tran, D. A., Hua, K. A., and Sheu, S. (2003). A New Caching Architecture for Efficient Video-On-Demand Services on the Internet. In *Symposium on Applications and the Internet*, pages 172–181. IEEE.
- Turrini, E. (2004). An Architecture for Content Distribution Internetworking. Technical report, Department of Computer Science, University of Bologna, Italy.
- Venkataramani, A., Dahlin, M., and Weidmann, P. (2001). Bandwidth Constrained Placement in a WAN. *ACM Principles of Distributed Computing*.
- Venkateswaran, R. (2001). Virtual Private Networks. *IEEE Potentials*, 20(1):11–15.
- Vuong, S., Gan, M., and Cai, X. (2002). A Hierarchical Request-Routing Architecture for Content Internetworking. In *MILCOM 2002*, volume 2, pages 946 – 951.
- Wang, B., Sen, S., Adler, M., and Towsley, D. (2002). Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In *INFOCOM*. IEEE.
- WebTVlist (2005). Live Internet TV Directory: www.webtvlist.com.

- Yang, M. and Fei, Z. (2003). A Model for Replica Placement in Content Distribution Networks for Multimedia Applications. In *IEEE International Conference on Communications, 2003. ICC 2003*, volume 1, pages 557–561. IEEE.
- Yin, J., Alvisi, L., Dahlin, M., and Iyengar, A. (2001). Engineering Server-driven Consistency for Large-scale Dynamic Web Services. In *Proc. 10th World Wide Web Conference*, Hong Kong.
- Zhao, B. Y., Kubiawicz, J. D., and Joseph, A. D. (2001). Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Technical Report UCB/CSD-01-1141, UC Berkeley.
- Zhu, Y., Wu, M., and Shu, W. (2003). Comparison Study and Evaluation of Overlay Multicast Networks. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 3.
- Zosin, L. and Khuller, S. (2002). On directed Steiner trees. In *Proc. the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*.

Appendix A

Lagrangian Multiplier Update Procedures

A.1 Introduction

In the Lagrangian Heuristics developed in this thesis, the Lagrangian multipliers are adjusted between heuristic iterations using a well known subgradient procedure, which aims to tune the relaxed problems in away that gradually increase the distance between the relaxed solution (i.e. lower bound) and the optimal solution of the original non-relaxed problems.

The specific subgradient formulae used to adjust the Lagrangian multipliers in each of the heuristics developed are provided below.

A.2 Adjusting Multipliers in the Web ODN Provisioning Problem

The following procedure describes how the multipliers are adjusted at each iteration of the heuristic:

1. Initialize:

Set $LB = 0$, $UB = \infty$, $t = 1$, $\mu_{ij}^{kt} = 0$

Note that the superscript t is added to represent the value of a particular pa-

parameter at iteration t

2. At iteration t :

Solve subproblems: $P1(\mu^t)$ and $P2(\mu^t)$

$$P_{LR}^t = P1(\mu^t) + P2(\mu^t)$$

Generate feasible solution based on solution r_{ij}^k of $P2$; resulted in objective function P^t

3. Update problem bounds:

$$LB = \max\{LB, P_{LR}^t\}$$

$$UB = \min\{UB, P^t\}$$

4. Update the multipliers:

$$\mu_{ij}^{k(t+1)} = \max\{\mu_{ij}^{kt} + \theta^t(r_{ij}^k - \lambda_{jk}x_{ik}), 0\}$$

where step size θ^t is calculated by:

$$\theta^t = \frac{\phi(1.05UB - P_{LR}^t)}{\sum_{i=1}^N \sum_{j=1}^M \sum_{k=1}^K (r_{ij}^k - \lambda_{jk}x_{ik})^2} \quad (\text{A.1})$$

and ϕ is a number between 0 and 2

$$t = t + 1$$

5. Go back to step 2

A.3 Adjusting Multipliers in the Pay-Per-View ODN Provisioning Problem

1. Initialize:

$$\text{Set } LB = 0, UB = \infty, t = 1, \mu_{mk}^t = 0, w_i^t = 0$$

2. At iteration t :

Solve subproblems: $Q_1(\mu^t, w^t)$ and $Q_2(\mu^t, w^t)$

$$Obj(Q_{LR})^t = Obj(Q_1(\mu^t, w^t)) - Obj(Q_2(\mu^t, w^t))$$

Generate feasible solution based on solutions of Q_1 and Q_2 , as described in Section 5.4.2, which gives an objective function value $Obj(Q_{eq})^t$

3. Update problem bounds:

$$LB = \max\{LB, \text{Obj}(Q_{LR})^t\}$$

$$UB = \min\{UB, \text{Obj}(Q_{eq})^t\}$$

4. Update Lagrangian multiplier:

$$w_i^{t+1} = \max\{w_i^t + \theta^t (\sum_{m \in V_a} \sum_{k \in K} \beta_k r_{mi}^k - y_i P_i), 0\} \quad (\text{A.2})$$

$$\mu_{mk}^{t+1} = \mu_{mk}^t + \theta^t (\sum_{i \in V_s} r_{mi}^k - u_k \lambda_{mk}) \quad (\text{A.3})$$

where step size θ^t is calculated by:

$$\theta^t = \frac{\phi [1.05UB - \text{Obj}(Q_{LR})^t]}{\sum_{i=1}^N (\sum_{m \in V_a} \sum_{k \in K} \beta_k r_{mi}^k - y_i P_i)^2 + \sum_{j=1}^M \sum_{k=1}^K (\sum_{i \in V_s} r_{mi}^k - u_k \lambda_{mk})^2} \quad (\text{A.4})$$

and ϕ is a number between 0 and 2

$$t = t + 1$$

5. Go back to step 2

A.4 Adjusting Multipliers in the Streaming ODN Provisioning Problem

1. Initialize:

$$\text{Set } LB = 0, UB = \infty, t = 1, \mu_{mi}^{k,t} = 0, w_{ij}^{kn,t} = 0$$

2. At iteration t:

$$\text{Solve subproblems: } S_1(\mu^t, w^t), S_2(\mu^t, w^t) \text{ and } S_3(\mu^t, w^t)$$

$$\text{Obj}(S_{LR})^t = \text{Obj}(S_1(\mu^t, w^t)) + \text{Obj}(S_2(\mu^t, w^t)) + \text{Obj}(S_3(\mu^t, w^t))$$

Generate feasible solution to original problem S , based on solutions of S_1 , S_2 and S_3 , as described in Section 6.3.3, which gives an objective function value $\text{Obj}(S^t)$

3. Update problem bounds:

$$LB = \max\{LB, \text{Obj}(S_{LR})^t\}$$

$$UB = \min\{UB, \text{Obj}(S^t)\}$$

4. Update Lagrangian multipliers:

$$w_{ij}^{kn,t+1} = \max\{w_{ij}^{kn,t} + \theta^t(f_{ij}^{kn} - t_{ij}^k), 0\} \quad (\text{A.5})$$

$$\mu_{mi}^{k,t+1} = \max\{\mu_{mi}^{k,t} + \theta^t(r_{mi}^k - x_{ik}\lambda_{mk}), 0\} \quad (\text{A.6})$$

where step size θ^t is calculated by:

$$\theta^t = \frac{\phi [1.05UB - \text{Obj}(S_{LR})^t]}{\sum_{m \in V_a} \sum_{i \in V_s} \sum_{k \in K} (r_{mi}^k - x_{ik}\lambda_{mk})^2 + \sum_{(i,j) \in E} \sum_{n \in V_s} \sum_{k \in K} (f_{ij}^{kn} - t_{ij}^k)^2} \quad (\text{A.7})$$

and ϕ is a number between 0 and 2

$$t = t + 1$$

5. Go back to step 2

A.5 Other Notes

In the subgradient procedures above, we have introduced some minor modifications compared to the standard formulae commonly seen in literature:

- The gap $(UB - LB^t)$ normally used in the numerator of formula (A.1), (A.4) and (A.7) has been replaced with $(1.05UB - LB^t)$, as suggested in (Glover and Laguna, 1993), which was observed in our experiments to speed up convergence at the final iterations.
- The step size constant ϕ are normally initialized to 2 and halved after some iterations that does not produce improve in problem bounds. The intuition is that by adjusting step size this way, the subgradient procedure will start off with larger step sizes and try to converge quickly to the optimal set of multiplier values, but when this does not produce improvements after a number of iterations, smaller step sizes will allow finer searches to be done. In some experiments, instead of halving ϕ , we also used a graceful reduction of $\phi = 0.8\phi$ after a number of iterations that does not improve the lower bound, thus

allow the heuristic to perform more searches at large step size. This was found to improve convergence speed in some experiments

Appendix B

Solving the Subproblems

This appendix presents the solution procedures for a number of subproblems created by relaxing ODN provisioning problems in Chapter 5 and 6.

B.1 Solving subproblems Q_1 (5.10) and Q_2 (5.11)

Both subproblems (Q_1) and (Q_2) can be solved easily by simple inspection. The solution to (Q_1) can be found as follows:

$$u_k = \begin{cases} 1 & \text{if } \sum_{j=1}^M \mu_{jk} \lambda_{jk} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.1})$$

Subproblem (Q_2), on the other hand, can be further decomposed into V_s subproblems, each corresponds to one separate server location $i \in V_s$. With subscript i dropped, the subproblem for location i is:

$$(\text{B.2}) \quad \text{Min} \quad \sum_{m \in V_a} \sum_{k \in K} r_m^k A_m^k + \sum_{k \in K} x_k H_k + yD \quad (\text{B.2})$$

subject to:

$$\begin{aligned} r_m^k (\tau_m - T_k) &\leq 0, \forall m, k \\ r_m^k &\leq \lambda_{mk} x_k, \forall m, k \\ x_k &\leq y, \forall k \end{aligned}$$

$$x_k, y \in \{0, 1\}; r_m^k \in [0, \lambda_{mk}]$$

For each subproblem (Q_{2-i}) , we need to consider only two possibilities:

1. if $y = 0$ (server i closed) then $x_k = 0, \forall k, r_m^k = 0, \forall m, k$ and $Obj(Q_{2-i}) = 0$, due to variable dependencies.
2. if $y = 1$ (server i opened) then the objective function becomes:

$$\sum_{k \in K} [\text{Min} (\sum_{m \in V_a} r_m^k A_m^k + x_k H_k)] + D$$

and (Q_{2-i}) can be solved through a series of K subproblems, with each corresponding to one content item $k \in K$. Subproblem corresponding to server location i , object k , with these subscripts dropped, is:

$$(Q_{2-i,k}) \quad \text{Min} \quad \sum_{m \in V_a} r_m A_m + xH \quad (\text{B.3})$$

subject to:

$$\begin{aligned} r_m(\tau_m - T) &\leq 0, \forall m \\ r_m &\leq \lambda_m x, \forall m \\ x &\in \{0, 1\}; r_m \in [0, \lambda_m] \end{aligned}$$

Each each problem $(Q_{2-i,k})$ can be solved using the following simple procedure:

1. Assuming $x = 1$, we have:

$$r_m = \begin{cases} \lambda_m & \text{if } \tau_m \leq T \text{ and } A_m \leq 0 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.4})$$

2. Using these variable values, the objective function of $Q_{2-i,k}$ can be calculated, if $Obj(Q_{2-i,k}) \leq 0$ then we have found the optimal solution, otherwise the optimal solution is $x = 0, r_j = 0, \forall j$ and $Obj(Q_{2-i,k}) = 0$

B.2 Solving subproblem S_3 (6.17)

As shown in Section 6.3.2, subproblem (S_3) (6.17) is equivalent to the following problem:

$$(S_{3eq}) \text{ Min } \sum_{(i,j) \in E} l_{ij} D_{ij} + \sum_{i \in V_s} y_i c_i$$

subject to:

$$l_{ij} \leq y_i, \quad (i, j) \in E$$

$$l_{ij} \leq y_j, \quad (i, j) \in E$$

$$l_{ij}, y_i \in \{0, 1\}$$

where $D_{ij} \leq 0$ and $c_i \geq 0$. This subproblem can be stated as follows: given a graph $G = \langle V, E \rangle$, where each node $i \in V$ has cost $c_i \geq 0$ and each edge $(i, j) \in E$ has cost $D_{ij} \leq 0$, select a set of nodes and edges that will result in the minimum total cost, subject to the constraint that an edge can only be selected if its end nodes also are selected.

We will solve this problem by relaxing it into a linear problem with continuous variables, as follows:

$$(S_{3L}) \text{ Min } \sum_{i \in V} y_i c_i + \sum_{(i,j) \in E} l_{ij} D_{ij} \tag{B.6}$$

Subject to:

$$l_{ij} \leq y_i, \quad i \in V, (i, j) \in E \tag{B.7a}$$

$$l_{ij} \leq y_j, \quad j \in V, (i, j) \in E \tag{B.7b}$$

$$y_i \leq 1, \quad i \in V \tag{B.7c}$$

$$l_{ij}, y_i \geq 0 \tag{B.7d}$$

We will now prove that by solving the relaxed problem above and then set the value of any non-zero decision variables to 1, an optimal solution to (S_{3eq}) would also be obtained.

Lemma 1 Consider a solution $\{l_{ij}^*, y_i^*\}$ that is optimal to (S_{3L}) , denote G^* as the subgraph consisting of nodes and edges selected by this solution, i.e. $G^* = \langle V^*, E^* \rangle$, where $V^* = \{i \in V | y_i^* > 0\}$ and $E^* = \{(i, j) \in E | l_{ij}^* > 0\}$. If $v \subset V^*$ is a subset of nodes that satisfy the conditions: $y_i^* = \alpha, \forall i \in v$ and $\min_{i \in \{V \setminus v\}} y_i^* = \beta \geq \alpha$, then $\{\tilde{x}_{ij}, \tilde{y}_i\}$ where

$$\tilde{x}_{ij} = \begin{cases} \beta & \text{if } (i, j) \in E^* \text{ and } i \in v \text{ or } j \in v \\ l_{ij}^* & \text{otherwise} \end{cases}$$

$$\tilde{y}_i = \begin{cases} \beta & \text{if } i \in v \\ y_i^* & \text{otherwise} \end{cases}$$

is also an optimal solution.

Proof:

- Since $\beta = \min_{i \in \{V \setminus v\}} y_i^*$, using this value for nodes in v and their incident edges does not create any constraint violations. Thus $\{\tilde{x}_{ij}, \tilde{y}_i\}$ is a feasible solution.
- Denote e the set of selected edges that are incident to nodes in v , i.e. $e \subset E^*$ and $i \in v$ or $j \in v, \forall (i, j) \in e$, and \tilde{C} the objective function value given by $\{\tilde{x}_{ij}, \tilde{y}_i\}$, we have:

$$\begin{aligned} \tilde{C} - C^* &= \sum_{i \in V^*} \tilde{y}_i c_i + \sum_{(i, j) \in E^*} \tilde{x}_{ij} D_{ij} - \left(\sum_{i \in V^*} y_i^* c_i + \sum_{(i, j) \in E^*} l_{ij}^* D_{ij} \right) \\ &= \sum_{i \in v} \tilde{y}_i c_i + \sum_{(i, j) \in e} \tilde{x}_{ij} D_{ij} - \left(\sum_{i \in v} y_i^* c_i + \sum_{(i, j) \in e} l_{ij}^* D_{ij} \right) \quad (\text{B.8}) \\ &= \sum_{i \in v} \beta c_i + \sum_{(i, j) \in e} \beta D_{ij} - \left(\sum_{i \in v} y_i^* c_i + \sum_{(i, j) \in e} l_{ij}^* D_{ij} \right) \end{aligned}$$

Since $\{l_{ij}^*, y_i^*\}$ is an optimal solution, we must have $\sum_{i \in v} y_i^* c_i + \sum_{(i, j) \in e} l_{ij}^* D_{ij} \leq 0$, otherwise unselecting all nodes in v and edges in e would result in a better solution than the optimal, which cannot happen.

Because $l_{ij}^* \leq y_i^* = \alpha, l_{ij}^* \leq y_j^* = \alpha, \forall i, j \in v, (i, j) \in e$ (due to constraints (B.7a), (B.7b)), and the fact that $D_{ij} \leq 0$, we have:

$$\begin{aligned}
0 &\geq \sum_{i \in v} y_i^* c_i + \sum_{(i,j) \in e} l_{ij}^* D_{ij} = \sum_{i \in v} \alpha c_i + \sum_{(i,j) \in e} l_{ij}^* D_{ij} \\
&\geq \sum_{i \in v} \alpha c_i + \sum_{(i,j) \in e} \alpha D_{ij} \\
&\geq \sum_{i \in v} \beta c_i + \sum_{(i,j) \in e} \beta D_{ij} \quad (\text{since } \beta \geq \alpha)
\end{aligned} \tag{B.9}$$

From (B.8) and (B.9), we have: $\tilde{C} - C^* \leq 0$. Since $\tilde{C} - C^* < 0$ does not happen, we must have $\tilde{C} = C^*$ and thus $\{\tilde{x}_{ij}, \tilde{y}_i\}$ is also an optimal solution.

Theorem 1 *If $\{l_{ij}^*, y_i^*\}$ is an optimal solution to (S_{3L}) , $\{l'_{ij}, y'_i\}$, where*

$$\begin{aligned}
l'_{ij} &= 1, \text{ if } l_{ij}^* > 0 \\
y'_i &= 1, \text{ if } y_i^* > 0 \\
l'_{ij}, y'_i &= 0, \text{ otherwise}
\end{aligned}$$

is an optimal solution to (S_{3eq}) .

Proof:

Denote G^* as the subgraph consisting of nodes and edges selected by solution $\{l_{ij}^*, y_i^*\}$, i.e. $G^* = \langle V^*, E^* \rangle$, where $V^* = \{i \in V | y_i^* > 0\}$ and $E^* = \{(ij) \in E | l_{ij}^* > 0\}$. The optimal objective function of (S_{3L}) is then:

$$C^* = \sum_{i \in V^*} y_i^* c_i + \sum_{(i,j) \in E^*} l_{ij}^* D_{ij}$$

Obviously we must have $C^* \leq 0$, since the trivial solution, where no nodes or edges are selected, already gives an objective function of 0.

We use the result in Lemma 1 to prove our theorem as follows. Without losing generality, assume $0 < y_1^* \leq y_2^* \leq \dots \leq y_N^*$, where $N = |V^*|$, and denote $v(n) = \{i \in V^* | 1 \leq i \leq n\}$ and $e(n) = \{(i, j) \in E^* | i \in v(n) \text{ or } j \in v(n)\}$. In other words, $g(n) = \langle v(n), e(n) \rangle$ is a subgraph of G^* , consisting of nodes $1, 2, \dots, n$ and their incident edges.

Consider subgraph $g(1)$, if we set the decision variables for nodes and edges in $g(1)$ to y_2^* , according to Lemma 1, the resulted solution is also optimal. More importantly, in this solution all nodes in $g(2)$ has the same value (y_2^*) which is smaller than those

of the remaining nodes. Thus we can again set the decision variables for nodes and edges in $g(2)$ to y_3^* and obtain another optimal solution, and so on. By iteratively adjusting the solution in this way (i.e. at iteration n ($1 \leq n \leq N - 1$), set values for nodes and edges in $g(n)$ to y_{n+1}^*) a final solution $\{\tilde{x}_{ij}, \tilde{y}_i\}$ is obtained, where $\tilde{x}_{ij} = \tilde{y}_i = y_N^*, \forall i \in V^*, (i, j) \in E^*$, which is also an optimal solution. Thus we have:

$$\begin{aligned}
 C^* &= \sum_{i \in V^*} \tilde{y}_i c_i + \sum_{(i,j) \in E^*} \tilde{x}_{ij} D_{ij} \\
 &= \sum_{i \in V^*} y_N^* c_i + \sum_{(i,j) \in E^*} y_N^* D_{ij} \\
 &\geq \sum_{i \in V^*} c_i + \sum_{(i,j) \in E^*} D_{ij} \quad (\text{since } C^* \leq 0 \text{ and } 0 < y_N^* < 1) \\
 &= \sum_{i \in V^*} y'_i c_i + \sum_{(i,j) \in E^*} l'_{ij} D_{ij}
 \end{aligned} \tag{B.10}$$

Since $\{l'_{ij}, y'_i\}$ is a feasible solution to (S_{3L}) , equation (B.10) means that it is also an optimal to this problem. Besides, since $\{l'_{ij}, y'_i\}$ is also feasible to problem (S_{3eq}) , it must be an optimal solution to this problem too \square .

Appendix C

Estimating Minimum Number of Replicas

Consider a graph $G = \langle V, E \rangle$, where V is the set of nodes and E is the set of edges. Two subsets of nodes are defined - access nodes V_a and potential server nodes V_s , where $V = V_a \cup V_s$, $N_a = |V_a|$ and $N_s = |V_s|$. Assume that the distance τ_{mi} between every access node m and every server node i is known, we need to find the minimum number of replicas that have to be placed so that each access node m has at least one replica within distance T . This could be found by solving the following optimization problem:

$$\text{Min} \sum_{i \in V_s} q_i \tag{C.1}$$

subject to:

$$\begin{aligned} \sum_{i \in V_s} z_{mi} &\geq 1, \forall m \\ z_{mi}(\tau_{mi} - T) &\leq 0, \forall m, i \\ z_{mi} &\leq q_i, \forall i, m \\ z_{mi}, q_i &\in \{0, 1\} \end{aligned}$$

where $q_i = 1$ indicates that a replica should be placed at server location i .