Enabling Performance Tradeoffs Through Dynamic Configuration of Advanced Network Services

A Thesis
Presented to
The Academic Faculty

by

Jinliang Fan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
December 2005

Enabling Performance Tradeoffs Through Dynamic Configuration of Advanced Network Services

Approved by:

Dr. Mostafa H. Ammar
College of Computing, Adviser

Dr. Ellen W. Zegura
College of Computing

Dr. Jun Xu
College of Computing

Dr. Henry Owen
School of Electrical and Computer Engineering

Dr. Constantinos Dovrolis
College of Computing

Date Approved: Nov 28, 2005

To my parents,

For all the sacrifices you have made as well as the love and the support you have always provided me.

ACKNOWLEDGEMENTS

I would first like to thank my dissertation committee members, Jim Xu, Constantinos Dovrolis, Ellen Zegura and Henry Owen, for all of their time and suggestions. I greatly appreciate their insight and comments that have influenced and improved my research.

I am so fortunate to be a part of the Networking and Telecommunication research group, from which I have received not only inspiration and technical support but also encouragement and friendship during the years. Paul Judge, Li Zou, Zongming Fei, and Shashidhar Merugu provided me with very useful feedback on my research on many occasions. Qi He, Taehyun Kim, Pradnya Karbhari and Donghua Xu offered a lot of help in job searching. All the members of the NTG group have played an important role in making my years at the GCATT building pleasant and unforgettable.

Beyond the Networking and Telecommunication research group, I want to say thanks to Leo Mark, Mostaque Ahamad, Umakishore Ramachandran, Yaxin Liu and Yuan Chen for their many advices leading me through my first year in Georgia Tech. I also want to thank Zhen Liu and Dimitrios Pendarakis for providing me the opportunity to work in IBM T.J. Watson Research Center where I developed part of the results forming this thesis. My fiance Tianying Chang is a big reason why I am writing this section of my dissertation today. In both research and life, her unselfish support followed me throughout my Ph.D. years and she brought much more fun to this long journey.

Finally, and most importantly, I want to express my sincere gratitude for my advisor, Mostafa Ammar, for his guidance, support, and patience throughout my Ph.D. years. He provided me with the flexibility to choose projects and steered me in the right direction through valuable feedback and constructive criticism. He taught me the value of perseverance as well as the value of vision, the value of action as well as the value of direction. I have certainly grown throughout this study due to his words of wisdom, encouragement and faith in me.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ACKNOWLEDGEMENTS . . . . . **iv**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

SUMMARY . . . . . **xi**

I INTRODUCTION **1**

1.1 Overview of Research Objectives 1

1.1.1 Service Overlay Network 3

1.1.2 Content Resiliency Service Networks 6

1.2 Research Methodology and Components 9

1.2.1 Research Methodologies 9

1.2.2 Research Summary 10

1.3 Thesis Organization 11

II BACKGROUND AND RELATED WORK **12**

2.1 Overlay Networks 12

2.1.1 Overview of Overlay Networks 12

2.1.2 Application Layer Multicast 13

2.1.3 Peer-to-Peer Networks 13

2.1.4 Infrastructural Overlay Networks 14

2.2 Network Topology Design 15

2.2.1 Network Design 15

2.2.2 Research Work in Network Topology 16

2.2.3 Overlay Topology Configuration 16

2.2.4 Dynamic Reconfiguration Policies 17

2.3 Content Resiliency Service 18

2.3.1 Content Replication and Content Resiliency 18

2.3.2 Modeling Correlated Failures 18

III STATIC TOPOLOGY CONFIGURATION FOR SERVICE OVERLAY NETWORK 20

3.1 Introduction 20

3.2 Problem Formulation 22

3.2.1 Static Overlay Topology Design Problem 23

3.2.2 An Integer Linear Programming Formulation 23

3.3 Optimization Techniques 25

3.3.1 Problem Complexity — Proof of NP-Hardness 25

3.3.2 A Heuristic Approach Based On Simulated Annealing 27

3.4 Performance 30

3.5 Summary 31

IV DYNAMIC TOPOLOGY CONFIGURATION FOR SERVICE OVERLAY NETWORK 33

4.1 Introduction 33

4.2 Problem Formulation 35

4.2.1 Dynamic Overlay Topology Design Problem 35

4.2.2 Topology Reconfiguration Policies 38

4.2.3 Methodology and An MDP Formulation 40

4.3 Properties and Structure of Optimal Reconfiguration Policies 42

4.3.1 Trends of Aggressiveness 42

4.3.2 Internal Structures of Optimal Reconfiguration Chains 46

4.3.3 Summary 49

4.4 Constructing Reconfiguration Policies for Large Systems 50

4.4.1 Always-Change Policy and Never-Change Policy 50

4.4.2 Cluster-Based Policies 51

4.5 Performance Of Approximate Policies 53

4.5.1 Performance of Constructed Policies — Small Networks 53

4.5.2 Performance of Constructed Policies — Large Networks 54

4.5.3 Implications on Overlay Network Design 57

4.6 Summary 60

V COST-EFFECTIVE CONTENT RESILIENCY SERVICE 61
5.1 Introduction 61
5.2 A General Approach for Modeling Correlated Failures 65
5.3 Static Configuration Problem in Content Resiliency Service 67
5.3.1 Failure Models 68
5.3.2 Analysis of Data Availability 70
5.3.3 Cost-Effective Configuration of Replication Locations 71
5.4 Optimization Algorithms for Static Configuration Problem 73
5.4.1 Branch-and-Bound 73
5.4.2 Properties of Optimal Solutions 74
5.4.3 Heuristic Algorithms Using Minimum Enclosing Circles 75
5.4.4 Performance of Heuristic Algorithms 77
5.5 Dynamic Configuration Problem in Content Resiliency Service 78
5.5.1 Problem Formulation 79
5.5.2 Dynamic Reconfiguration Policies 80
5.6 Summary 84
VI CONTRIBUTIONS OF THE THESIS AND FUTURE WORK 86
6.1 Summary of Contributions 86
6.2 Future Work 89
6.2.1 Dynamic Configuration of Sensing Overlay Networks 89
6.2.2 Extension to The Study of Content Resiliency Service 91
REFERENCES 95
VITA 102

LIST OF TABLES

1 Experiment Parameters 54

LIST OF FIGURES

1 Service Overlay Network 3

2 Factors in Overlay Topology Design 5

3 Routing Data Through Overlay Networks 5

4 Content Resiliency Service 7

5 Choosing Replication Locations from Pool of Content Servers 8

6 Methodology and Components of Research 9

7 Pseudo-code of Finding Optimal-Static Topology Using Simulated Annealing 28

8 Mutating Operation 2 29

9 Mutating Operation 3 29

10 Performance of Subroutine for Finding Static-Optimal Topology. Note that in this experiment the "Optimal" line and the "Simulated Annealing" line in (a) exactly overlap. 30

11 Flow Disturbance Caused by Topology Reconfiguration 36

12 $X(t)$ is a continuous Markov process 41

13 Various Policies Make Different Decision at State $<\mathcal{C}_1, \mathcal{T}_1>$ 41

14 Communication Requirement Transition Model With Two Communication Patterns 42

15 Optimal Policies. Solid lines and dashed lines show how the overlay topology is reconfigured when the communication pattern changes from $\mathcal{C}_1$ to $\mathcal{C}_2$ and from $\mathcal{C}_2$ to $\mathcal{C}_1$, respectively. 43

16 Optimal Reconfiguration Chains. Solid lines and dashed lines have the same meanings as in Fig. 15. 43

17 Affect of Transition Rates 45

18 Communication Requirement Transition Model with 6 Communication Patterns 45

19 Trends of Aggressiveness and Costs 46

20 Optimal Reconfiguration Chains. Solid lines represent transitions between states with different overlay topology, while dashed lines represent transitions between states with the same overlay topology. 47

21 Pseudo-code of Constructing Cluster-Based Polices 52

22 Effect of Given Number of Clusters 55

23 Performance of Approximate Policies Against Optimal Policies 55

24 Performance of Approximate Policies Against Optimal Policies in Non-Markovian Cases 55
25 Using Other Formulations of Reconfiguration Cost 57
26 Degree of Overlay Networks versus Cost of Reconfiguration Policies 58
27 Geographically Correlated Failure Model 68
28 Optimal Backup Sets 75
29 Performance of Optimization Algorithms 77
30 ECP: Availability, Communication Cost and Reconfiguration Cost 81
31 IATP: Communication Cost and Reconfiguration Cost Affected by Threshold Ψ 83
32 IATP: Overall Cost Affected by Threshold Ψ and Weighting Factor β . . . 84
33 Sensing Overlay Network 90

SUMMARY

Configuration capabilities are important for modern advanced network services. Network conditions and user populations have been significantly diversified after decades of evolution of the Internet. Configuration capabilities allow network services to be adapted to spatial, temporal, and managerial variations in application requirements and service operation conditions.

Network service providers need to decide on the best configuration. Ideally, a network service should have all of its components optimally configured to most effectively deliver the functionality for which it was designed. The "optimal" configuration, however, is always a compromise between different metrics. To decide on an optimal configuration, the prominent performance and cost metrics must be identified, modeled, and quantified. Optimization objective functions and constraints that combine these metrics should be formulated and optimization techniques should be developed. More important, in the scenarios where the application requirements and system conditions change over time, the service configuration needs to be dynamically adjusted and strategies that guide the reconfiguration decisions need to be developed. Because the actual process of configuring a network service incurs configuration costs, an optimal reconfiguration strategy should be one that achieves a tradeoff between the (re)configuration costs and static optimization objectives. Furthermore, such tradeoffs must be based on the consideration of long-term benefits instead of short-term interest.

This thesis focuses on understanding the strategies for dynamic (re)configuration of advanced network services positioned above the Transport Layer. Specifically, this thesis investigates the configuration and more important dynamic reconfiguration strategies for two

types of advanced network services: Service Overlay Networks, and Content Resiliency Service Networks. Unlike those network services whose configuration involves mainly arrangement of hard-wired components, these network services have the ability to change service configuration in small time scales. This makes the modeling of application requirements and system condition dynamics not only possible but also meaningful and potentially useful. Our ultimate goals in conducting the research presented in this thesis are to first develop modeling and optimization techniques for network service configuration and dynamic reconfiguration policies. We also seek to understand how effective techniques can improve the performance or reduce the cost of these advanced network services, thus demonstrating the advantage of allowing configurability in these advanced network services.

CHAPTER I

INTRODUCTION

1.1 Overview of Research Objectives

Over the decades, the Internet has evolved from a simple best-effort communication platform to an infrastructure that delivers a wide range of services to end users. Today, new services continue to emerge over the Internet and draw new users. Many of these services will flourish and generate new demands that drive the Internet to evolve into its next generation.

Modern advanced network services are typically designed with some configuration capabilities, for example, the capability of changing network topologies, tuning protocol parameters, reforming clusters, and adjusting management hierarchies of entities. Configuration capabilities are important for modern advanced network services because network conditions and user populations have been significantly diversified over the years. They give network service providers more flexibility in performance and cost tuning and allow the network services to be adapted to spatial, temporal, and managerial variation in operating conditions and user requirements.

While configuration capabilities give a network service provider the freedom of choosing among many options of configuration, decisions still need to be made on which configuration is the best option. Ideally, a network service should have all of its components configured to most effectively deliver the functionality for which it was designed. The "optimal" configuration, however, is always a compromise between different metrics, e.g., performance metrics such as latency and throughput, and cost metrics such as bandwidth consumption and CPU loads. Depending on the specific context in which the network service is provided, performance and cost metrics that come under consideration could vary among end-to-end latency of data delivery, network traffic incurred in the system, degree of load balancing and fairness among different entities, resiliency of the service under failures, security assurance of application data, and many others. To determine an optimal configuration, the prominent

performance and cost metrics must be identified, modeled, and quantified. Optimization objective functions and constraints that combines these metrics should be formulated and optimization techniques should be developed.

Tradeoffs also lie in the control dimension. The actual process of configuring a network service incurs configuration costs, e.g., extra control overhead or negative impact on the performance of the service. Such a configuration cost is often a concern of the protocol designers and this implies the configuration cost may need to be taken into account during configuration optimization. The issue is especially complex in the scenarios where the application requirements and system conditions change over time. A strategy that guides the reconfiguration decision needs to developed. Ideally, when the application requirements and system conditions change over time, the configuration of a network service should continue to be updated to perfectly match the current application requirements and system conditions. However, such a strategy may not be the optimal one when the (re)configuration cost is a concern. Updating service configuration too frequently may incur too much reconfiguration cost that even over shadow the benefit of the reconfiguration. A better reconfiguration strategy is one that achieves a good tradeoff between the (re)configuration cost and static optimization objectives, and such tradeoffs must be based on the consideration of long-term benefit instead of short-term interest.

This thesis seeks to understand the strategies of dynamic (re)configuration of advanced network services positioned above the Transport Layer. Unlike those network services whose configuration involves mainly arrangement of hard-wired components, these network services often have the capability of changing service configuration in short-time scale. This makes the modeling of application requirements and system dynamics not only possible but also meaningful and potentially useful. Our ultimate goals in conducting the research presented in this thesis are not only the development of modeling and optimization techniques for network service configuration and dynamic reconfiguration policies, but also understanding how these techniques can improve the performance or reduce the cost of these advanced network services and verifying the advantages of high configurability built in these services in the first place.

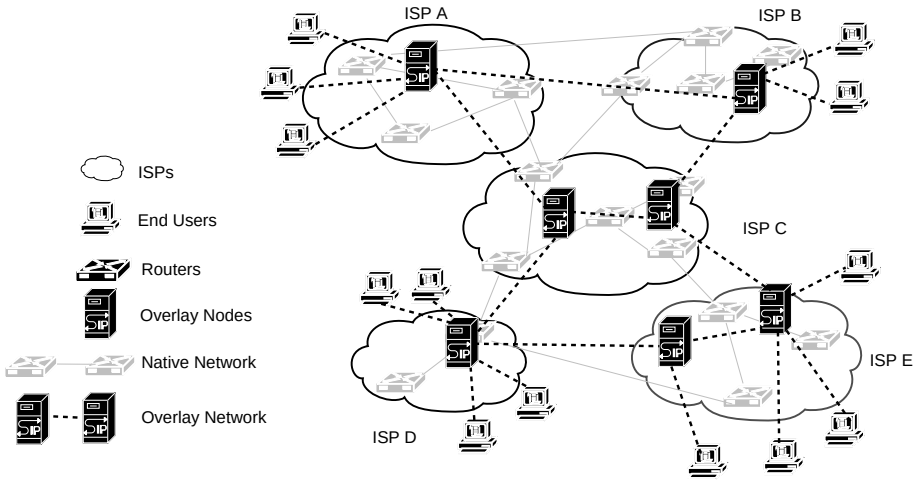


Figure 1: Service Overlay Network

Specifically, this thesis investigates the optimization of service configurations and, more importantly, dynamic reconfiguration strategies for two types of advanced network services: Service Overlay Networks, and Content Resiliency Service Networks.

1.1.1 Service Overlay Network

While the Internet has been fully commercialized and evolved into a ubiquitous medium of communication, its native routing infrastructure has become resistant to fundamental changes. This hinders the development of new network functionality (e.g., multicast, QoS) that heavily rely on such fundamental changes. The use of *overlay networks* has been proposed as an alternative solution that can provide the desirable flexibility and control of the routing infrastructure [70, 86, 32, 9]. Researchers have successfully used overlay networks to solve problems in various areas. For example, overlay networks have been employed to implement application layer multicast [20, 7, 68, 59], provide testbeds for new technologies [33, 28, 69], circumvent BGP faults and constraints [2], and provide countermeasures to DoS attacks [48].

While researchers study overlay networks in various contexts, in this thesis, we are most interested in *service overlay networks* [26, 2, 89] that are deployed and maintained by *overlay network providers*. As shown in Fig. 1, to build a service overlay network, an overlay

network provider deploys a number of specially designed *overlay nodes* across the Internet. On one hand, as a third-party service provider, an overlay network provider contracts with underlying ISPs and buys network bandwidth between these overlay nodes. On the other hand, the overlay network provider provides value-added network services to users at the end-systems, who access the overlay networks through one of the overlay nodes. Traffic between end-systems is carried by and routed through the overlay networks instead of the native networks.

One of most important issues in deploying a service overlay network is the configuration of its overlay topology. Positioned between the native network and the ultimate end users, an overlay topology constructed in favor of both could significantly improve the service performance or reduce the cost of operating the service. Consider the four-node example overlay network shown in Fig. 2. Fig. 2.a characterizes the service agreement between the overlay network provider and the underlying ISPs; the labels on the dashed lines show the operation costs of the potential overlay links between every pair of nodes — the price ISPs charge the overlay network provider for shipping one unit of data over these overlay links. Fig. 2.b shows a snapshot of the communication requirements aggregated over all the users at the end-systems; the labels on the lines (or the thickness of the lines) denote the data rates between every pair of overlay nodes, on behalf of the end users. Fig. 2.c shows a candidate overlay topology (Topology A); each edge denotes an overlay link. Provided that the overlay network ships data over the overlay paths that incur the minimum operation cost, the flow of data is shown in Fig. 3.a. Fig. 3.b presents the flow of data if another topology (Topology B) is adopted instead. In terms of operation cost, Topology B is not as good as A — although some data can now take a lower-cost path, a higher volume of data have to take a higher-cost path. Generally, the operation cost of an overlay varies when it runs on different overlay topologies and this raises the problem of finding the overlay topology that minimizes the operation cost for given communication requirements.

If the communication requirement is constant over time, the optimal configuration of overlay topology is static. If communication requirements change over time, for example, from the ones shown in Fig. 2.b to the ones shown in Fig. 3.c, the overlay topology may

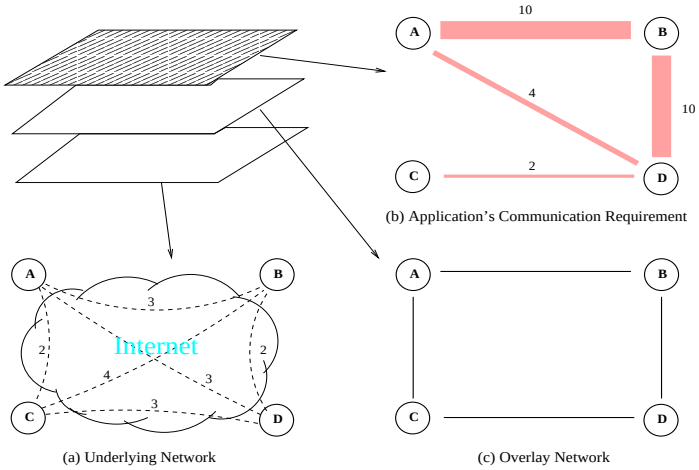
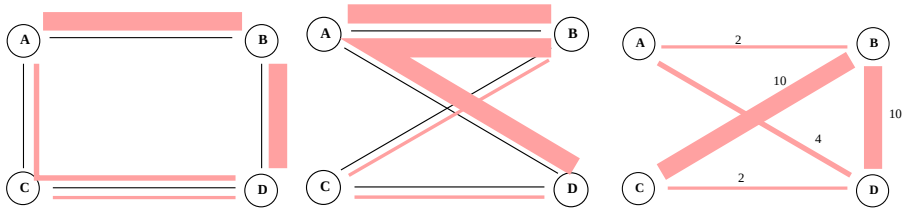


Figure 2: Factors in Overlay Topology Design



(a) Routing Over Overlay Topology A

(b) Routing Over Overlay Topology B

(c) Another Pattern of Communication Requirements

Figure 3: Routing Data Through Overlay Networks

need to be reconfigured. For overlay networks with tens of overlay nodes, it is feasible for the providers to monitor and statistically model the communication requirements in the system, and the topology reconfigurability allows an overlay network to be tuned dynamically when the communication requirements change [85, 87]. A question needs be answered, however, namely, when and how the overlay topology should be reconfigured under dynamic communication requirements. While overlay topology is reconfigurable in small time scale, changing overlay topology is not cost-free; it may incur both management overhead as well as potential disruption of end-to-end flows: overlay links need to be established or torn down; routing tables need to be updated; data in transit may get lost, delayed, or erroneously routed. In the presence of these costs, an *overlay topology reconfiguration policy*, which guides the topology selection any time the communication requirements change, needs to be designed and methods of constructing such a policy need to be developed.

1.1.2 Content Resiliency Service Networks

While engineers have invested a significant amount of effort in improving the reliability of computer hardware and software, failures are still common in today's computer systems and networks. Failures can be caused by not only worn-out devices and software bugs but also external factors such as power outage, environmental conditions and operational accidents. When the computer and network system that stores a piece of data fail, the data becomes unavailable to its users until the system is recovered. Replicating the same content to multiple content servers (e.g., data centers) at different locations is one of the most important strategies for improving the resilience of content in the presence of failures[35][90][25]. It can significantly improve the availability of the data even when multiple servers fail at the same time, and reduce the potential damage caused by the loss of data.

While content servers can be deployed by an organization for its private use, in this thesis we are most interested in *Content Resiliency Service Networks*, where a pool of content servers deployed by a third-party provide resiliency for service users by replicating data uploaded by the users. As shown in Fig. 4, a content resiliency service provider deploys a pool of tens or hundreds of content servers at different locations over the Internet. Users

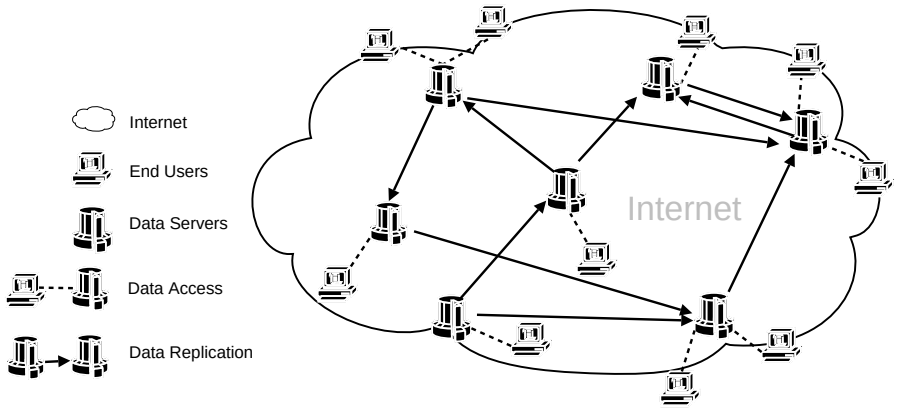


Figure 4: Content Resiliency Service

at the end-systems access (i.e., upload to and download from) their data at a local content server. The local data servers are then responsible to replicate the data to some of the other content servers in the pool so that the data is available when the local content servers or even part of the replicating content servers have failed at the same time. One one hand, a content resiliency service provider contracts with service users in the form of service agreement, which allows the users to specify the desired level of content resiliency or the penalty for loss of data. On the other hand, the content resiliency service provider pays *prevention costs*, such as the communication cost of delivering data from its origination local content server to the remote replicating content servers, the storage cost of storing the data, and types of management costs, to maintain the normal operation of the pool of content servers and achieve the agreed content resiliency.

Similar to the case of service overlay networks, there are configuration and dynamic reconfiguration issues in content resiliency service networks, particularly, the choice of replication locations for a piece of data from the pool of available content servers. Fig.5 shows a content resiliency service network with a pool of nine content servers, represented by the circles. A user, represented by the square box, uploads its data to its local content server S_0, and S_0 is responsible to choose a set of remote replication locations from the other eight content servers and deliver the data to the chosen set of servers. For example, it can choose servers S_3 and S_7 and form a replication set of $\{S_0, S_3, S_7\}$, or choose servers S_2 and S_5

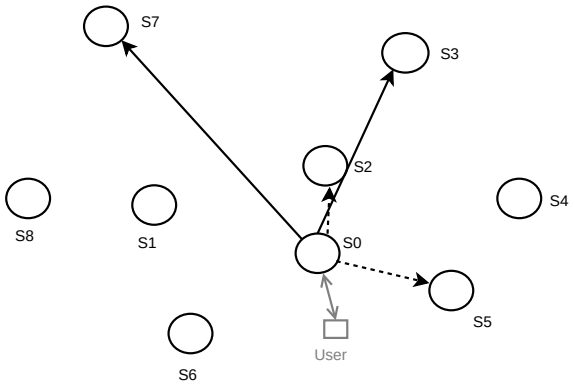


Figure 5: Choosing Replication Locations from Pool of Content Servers

and form a replication set of $\{S_0, S_2, S_5\}$. Intuitively, the further the servers replicating the same data reside from the content origination and from each other, the more unlikely the servers will fail in the same period but the more communication cost will be incurred. A good configuration of content server replication relationship should maintain a target level of content resiliency while minimizing the prevention cost incurred by the configuration and avoid excessive costs caused by over-reaction to potential risks. Furthermore, when content servers actually fail and recover, the replication sets need to be dynamically adjusted to maintain the desired level of content resiliency.

One critical component in the development of such a cost effective configuration is the assessment of failure risks for a given environment. The probability that all the content servers in the replication set fail at the same time must be quantified. When failures are local to each content server and therefore independent across the servers, such quantification is relatively easy to achieve. However, in contrast to many researchers' assumption that failures are independent from each others, many failures (e.g., those caused by power-outages, weather and environmental disasters, and worms) are correlated to each other. Though researchers have developed various approaches for modeling correlated failures [90, 5, 65], these models do not allow for deriving the probability of simultaneous failures for different subsets of entities in a practical, consistent and intuitive way. A new approach for modeling correlated failure needs to be develop before the configuration and dynamic reconfiguration issues in content resiliency service networks can be tackled.

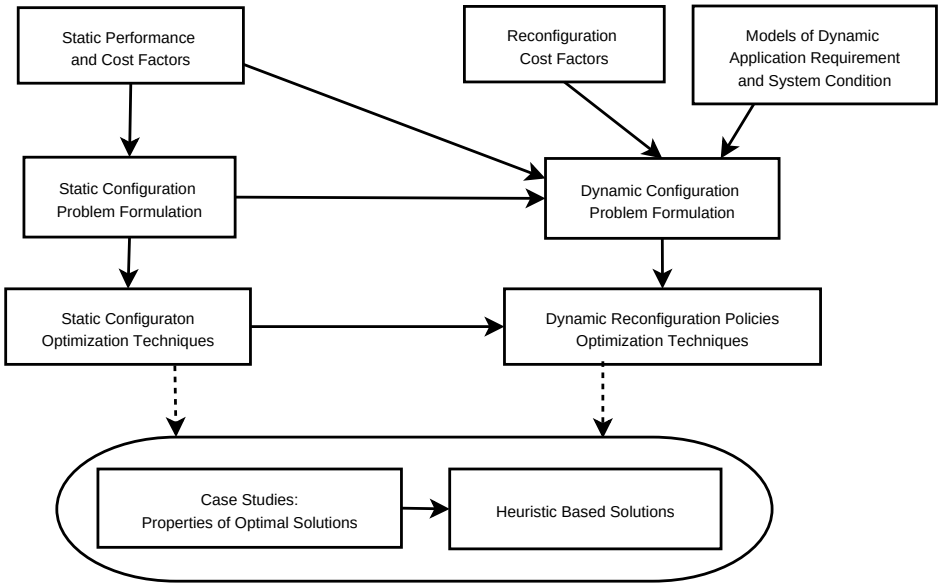


Figure 6: Methodology and Components of Research

1.2 Research Methodology and Components

1.2.1 Research Methodologies

Our methodology for studying static and dynamic configuration problems is shown in Fig. 6, which we commonly apply to both the study of service overlay networks and that of content resiliency service networks. For the static configuration problem, system configurable components are first identified, and static performance and cost factors that are most significantly affected by the choice of configuration are identified and formulated. The problem of choosing the optimal static configuration is then formulated as, typically, a constrained optimization problem, with a static objective function that combines different performance and cost factors. Finally, optimization techniques are developed to solve this static configuration optimization problem and the quality of solutions is evaluated. Furthermore, for the dynamic configuration problem, the cost factors in the process of (re)configuration are also identified. A model characterizing the dynamics of application requirements and system conditions is also necessary; without such a model, the development of dynamic reconfiguration policies is not meaningful. The problem of choosing the optimal dynamic reconfiguration policy is then formulated as optimizing the long-term average of a dynamic objective function that combines both the static objective function and the reconfiguration

costs. Finally, optimization techniques are developed to solve this dynamic configuration optimization problem and the quality of solutions is evaluated, during which the previously developed static configuration optimization techniques may be adopted as subroutines if appropriate.

Part of the major challenges in our research lie in the development of optimization techniques once the static and dynamic optimization problems are formulated. In the context of service overlay networks or content resiliency service networks, these problems are essentially combinatorial optimization problems — there are an exponential number of configuration options and an even larger number of dynamic reconfiguration policies. While optimization techniques for finding the real optimal solutions can be developed for these problems, their computational complexity renders them impractical for most of the system setups. Our methodology for addressing this situation is as follows. First, we do case studies using special cases (e.g., system setups with small number of entities) that are practically solvable using real optimization techniques, and observe the properties (e.g., structures) of the resulting optimal policies. We then use our observations as heuristics to assist the development of heuristic-based techniques that can generate solutions approximating the optimal ones for more general cases. Finally, the effectiveness of the heuristic-based techniques are verified through evaluating the quality of the approximate solutions.

1.2.2 Research Summary

The specific components of this thesis are summarized as follows:

Configuring overlay network topology for static demands: This thesis formulates the problem of finding the optimal overlay topology that can minimize the overall static operation cost incurred in a service overlay network for given static communication requirements. It presents a proof for the NP-hardness of the problem. It presents an integer linear programming model of the problem. It also presents an approximation algorithm based on simulated annealing and topology perturbation that is applicable to service overlay networks with a larger number of nodes.

Configuring overlay network topology for dynamic demands: This thesis presents

the overhead involved in the reconfiguration of overlay topology by analysis as well as experiments over Planet-Lab. It formulates the problem of finding optimal topology reconfiguration policies for given dynamic communication requirements. It presents a Markov decision process model for the problem and conducts a systematic study of the properties of optimal reconfiguration policies. It also presents heuristic methods for constructing different flavors of reconfiguration policies, i.e., never-change policy, always-change policy and cluster-based policies, to mimic and approximate the optimal ones. The policy construction methods help the study of other overlay design problems.

Configuring replication relationship in content resiliency service network This thesis presents a new approach for modeling correlated failures. It applies the approach to the case of geographically correlated failures. It formulates the problem of finding the optimal set of content server locations for given risk models. It presents both a branch-and-bound algorithm and minimum-enclosing-circle based heuristic algorithms for solving the problem. It also studies the cost and benefit of dynamic reconfiguration of content server replication relationship and presents an incremental and threshold-based reconfiguration policy.

1.3 Thesis Organization

The rest of this thesis is organized as follows. Chapter II discusses existing work related to the topics in this thesis. Chapters III and IV are dedicated to the study of topology configuration and reconfiguration issues in service overlay networks. In Chapter III, we investigate the problem of configuring overlay network topology for applications with *static* demands. In Chapter IV, we investigate the problem of configuring overlay network topology for applications with *dynamic* demands. Chapters V is dedicated to the study of replication relationship configuration and reconfiguration for content resiliency service network. In the chapter, we present a new approach for modeling correlated failures. With the assistant of such a model, we discusses the problem of configuring replication relationship between content servers under static and dynamic system conditions. In Chapter VI, we summarize the contributions of our research and discuss a few directions for the future work.

CHAPTER II

BACKGROUND AND RELATED WORK

This thesis investigates the optimization of service configurations and, more importantly, dynamic reconfiguration strategies for two types of advanced network services: Service Overlay Networks, and Content Resiliency Service Networks. This chapter provides an overview of the background and related work on the topics of this thesis.

2.1 Overlay Networks

2.1.1 Overview of Overlay Networks

In the most general terms, an overlay network is a computer network which is built on top of another network. The overlay architecture allows an extra layer of logic to be added between the entities in the network without changing the infrastructure of the underlying network. The overlay architecture can be seen in many parts of the Internet — the Internet itself was developed as an overlay over the telephone systems and cable systems about a couple decades ago.

In a narrower definition, which we use in this thesis, an overlay network usually refers to one that is built over the Internet IP networks and is connected by virtual or logical links established over the Transport Layer. In overlay networks, specially designed overlay nodes that are distributed over the Internet form another layer on top of the underlying Internet routing substrate. The nodes cooperate with each other to forward data on behalf of any pair of communicating nodes in the overlay network; each does not only generate and sink but also forwards traffic.

From the perspective of the locations where overlay nodes reside in the Internet native networks, overlay networks can be roughly classified into two categories: ones that involve only end-systems at the boundary of the Internet(i.e., DSL and Cable access networks), and ones that involves also entities (e.g., specially designed overlay routers) in the internal part

of the Internet (e.g., ISPs and backbone operators).

From the administrative perspective, overlay networks can be roughly classified into two categories too: ones that are formed voluntarily by ultimate end users who seek functionality that is not provided by the native network operators, and infrastructural ones deployed by a single organization or an association of organizations for private use or for value-added services.

From the perspective of their purpose, overlay networks can also be roughly classified into two categories: ones that are built mainly for traffic routing purposes, and ones that also combine other application layer functionality, such as security and information directory services.

Note that none of the above classifications is in strict sense. Many overlay networks fall at the boundary between two categories.

2.1.2 Application Layer Multicast

Overlay networks have been extensively used in *application layer multicast* protocols and projects, e.g., NICE[8], SplitStream[16], SCRIBE[17], Narada [21], Scattercast[19, 18],Yoid[34], Overcast[46], ALMI[68], HMTP[93], Bayeux[95] and HyperCast[59]. Most of them, except for a few (e.g., Overcast and Scattercast) that use overlay networks formed by strategically placed nodes around the whole Internet, are formed purely by the end-systems (i.e., the receivers of the multicast data) and are therefore also termed as *end-system multicast*. As an alternative for network layer multicast mechanisms, which failed to be widely adopted in the past decade due to various concerns and difficulties in its deployment, application layer multicast is proposed as new mechanism of sending the same copy of data from a source to a group of receivers and the flexibility of overlay networks make the deployment of application layer multicast over the Internet much easier than that of network layer multicast.

2.1.3 Peer-to-Peer Networks

The development of peer-to-peer technologies are closely related to the research of the overlay networks, especially those formed over voluntary end-systems. In a peer-to-peer system, a peer acts as both a client and a server, unlike the central server based systems where clients

and servers are clearly distinguished. The popularity of file sharing applications, most notably Kazaa[42], Gnutella[23], Freenet[22], Napster[43], EDonkey[39] and BitTorrent[12], has drawn intense interest in designing and studying peer-to-peer networks[66]. For example, CAN[76], Chord[63], Tapestry[94] and Pastry[81] studied the use of distributed hash tables to introduce structures in peer-to-peer networks. The computational and storage resource at the large number of end-systems, however, extend its usage beyond the scope of file sharing. By today, peer-to-peer networks have been proposed for numerous applications, e.g., content streaming[16], voice IP[44] and cooperative web cache relaying[67]. Peer-to-peer networks naturally use overlay networks that are formed over voluntary end-systems to disseminate queries and content.

2.1.4 Infrastructural Overlay Networks

While the Internet has been fully commercialized and evolved into a ubiquitous medium of communication, its fundamental routing infrastructure has become resistant to fundamental changes. This hinders the development of many new applications and technologies that rely heavily on such fundamental changes. Recently, there is intense interest in the community[70, 86, 32, 9] to the use of infrastructural overlay networks as a general solution that provides the desirable flexibility and control of the routing infrastructure and bridges the gap between the ossified fundamental infrastructure of the Internet and the applications running on top.

In practice, infrastructural overlay networks have been used to provide testbeds for new technologies. Examples are 6Bone[33], MBone[28] and PlanetLab[69]. They have also been used to find routing paths that circumvent BGP faults and constraints in the Internet native substrate[2, 78], deliver multicast traffic[46, 19], and support Quality of Service [54, 56]. The use of infrastructural overlay networks has stretched beyond purely providing network functions. For example, overlay networks are proposed to provide countermeasures to DoS attacks [48]. In a weaker definition, some content distribution network [52] and even the network of DNS servers can be considered as applications of infrastructural overlay networks.

Service overlay networks, which we focus on in this thesis, refer to the infrastructural

overlay networks that are deployed and maintained by third-party service providers and provide value-added network services to network users[26, 2, 89]. As third-party service providers, on one hand, these overlay network providers buy network resource from underlying network providers, and on the other hand, these overlay network providers provide value-added network services to end-systems. Traffic between end-systems is carried by and routed through the overlay networks instead of the native networks.

2.2 Network Topology Design

2.2.1 Network Design

Network design has played an important role in the development communication and computer networks. It deals with demand estimates, infrastructural investment and network operations and has much to contribute in optimizing the performance of a service and a service providers's operational expenditure. Network design issues virtually exist in all kinds of networks, though in different forms, e.g., intra-domain and inter-domain traffic engineering for IP networks[3] and virtual path design in ATM networks[36, 38]. When new technologies and new types of networks are deployed, it would not be surprising that the network design problems are revisited. The high deployment and maintenance cost of networks has stimulated the interest in developing efficient design models and optimization techniques.

Network design touches a wide scope of issues, such as routing, topology, fairness, resiliency, and capability reservation. Network design problems can be modeled as optimization problems that are constrained or unconstrained, linear or non-linear. Optimization techniques can be linear programming (e.g., Simplex, Interior Point Methods), mix-integer programming (e.g., Branch and Bound, Cutting-Plane), stochastic heuristics methods (e.g., Simulated Annealing, Evolutionary Algorithm), and other methods that take advantage of the special properties of the problem formulation and optimal solutions. Most of these optimization problems are NP-hard, so heuristic solutions play a significant role in network design. Excellent compilation of problems and solutions in network design can be found

in [72, 49]. Although the approach for problem modeling and the pool of available optimization techniques are similar for different network design problems, they are rarely exactly the same and non-trivial effort is required for customizing current optimization techniques or developing new algorithms that take advantage of the special properties of a problem formulation and its optimal solutions.

2.2.2 Research Work in Network Topology

Topological design problems form one of the most important categories of network design problems. They are closely related to location design problems, which concern most with the location of network nodes (e.g., router, hubs, access points). Topological design problems focus on determining the links in network for a given set of nodes and for certain network demand. There have been extensive research work (e.g., [4, 62, 27, 55]) over a wide variety of topological design problems.

Previous research work in network topology is not limited to topological design. Internet started to be commercialized and exploded in 1990's, after which getting topology information about the Internet becomes more and more difficult. A significant amount of effort of the network measurement community has been devoted to understanding the topology of the Internet[13, 29]. There are also important works in modeling the Internet topology and developing network topology generators[15, 92, 47].

2.2.3 Overlay Topology Configuration

Overlay topologies for application layer multicast have been extensively studied in the literature. In tree-based application layer multicast, receivers construct a tree topology and this tree topology is also used as the multicast tree for data delivery . Examples are Yoid[34], ALMI[68] and HMTP[93]. In mesh-based approach, receivers first construct a mesh overlay topology and then generate multicast trees over the mesh topology to deliver data. An example is Narada[21]. There are also protocols use an overlay topology that has multiple hierarchies. An example is NICE[8].

Most research in application layer multicast focus on the maintenance of the logic multicast tree and rarely considers the characteristics of underlying network topologies. In[77],

the authors proposed to build overlay topologies that are congruent with the underlying IP-level topologies to improve the performance of the overlay networks.

For infrastructural overlay networks with more general communication models, Li and Mohapatra[57] studied the impact of topology on overlay routing service. The authors compared several popular topologies (e.g., full-mesh, k-minimum-spanning-tree, mesh-tree, adjacent-connection, topology-aware-k-minimum-spanning-tree) and found that the overlay topology has significant impact on the overlay routing in terms of routing performance and overhead. They also found that the full mesh topology does not always give the best performance. Our work differs from theirs in the sense that we are interested in constructing overlay topologies specially customized to specific communication requirements rather than choose from several popular topologies.

The overlay topology construction method developed in this thesis is based on simulated annealing and topology perturbation. Simulated annealing is a global optimization meta-algorithm based on an analogy with the physical process of annealing [61, 71, 51]. It is a stochastic meta-heuristic algorithm that needs to be customized to specific optimization problems and its general structure has been applied to many combinatorial problems with good results [45, 50]. The application of topology perturbation in refining overlay topologies also has root in the literature of topological design for native networks ([82, 11]).

2.2.4 Dynamic Reconfiguration Policies

The policy issues in dynamic topology reconfiguration have been previously studied for optical networks. Rouskas and Ammar [80] studied how to dynamically update the connectivity of multihop WDM optical networks by tuning optical transmitters and receivers to accommodate traffic demands that vary over time. Baldine and Rouskas studied the dynamic reconfiguration for broadcast WDM optical networks to ensure that the traffic load remains balanced across the WDM channels under changing traffic conditions [6]. Our methodology of tackling the dynamic reconfiguration problem is similar to theirs: the problem of finding optimal reconfiguration policy is first modeled and solved as a Markov decision process [40] for small and Markovian systems, the properties of the optimal reconfiguration policies are

observed and then used as heuristics to help construct policies for the large systems.

Researchers have showed that it is feasible to monitor and infer the pattern of end-to-end communication in the overlay network and adjust the topology accordingly [85]. The mechanism of maintaining overlay topology varies in different research works but there is rare systematic study on the control overhead of overlay topology reconfiguration and its impact on the data routed in the overlay network and the applications running on top.

2.3 Content Resiliency Service

2.3.1 Content Replication and Content Resiliency

Resiliency issues have been extensively studied in almost every area of computer science, for example, fault tolerance of computer system[88], resilience of storage systems[74][79], and network resiliency[64].

Replication is one of the important resiliency strategies. By providing multiple identical instances of the same content at different system locations, the data can still be available and the system can continue operating properly in the event that some of its parts fail. Replication introduces redundancy. The overhead of maintaining multiple contend servers for resiliency purpose has been discussed in [10][60]. The discussion, however, is based on the assumption that failures are independent to each other.

Though content replication is an important mechanism to provide content resiliency, this is not its only use. Proactive content replication and reactive content caching have been among the major techniques for improving the performance of content services [52].

2.3.2 Modeling Correlated Failures

The impact of geographically correlated failures on resiliency has been studied in the literature, in both industry and academia, and solutions relying on remote placement of system components have been proposed (e.g., [35][90][25]). While such efforts are helpful in improving overall system resiliency, they suffer from the lack of good models for geographically correlated failures. Without such models, the improvement in system resiliency cannot be fully quantified and measures of cost efficiency cannot be developed.

The fundamental objective of modeling correlated failures is to provide the capability of calculating the probability of simultaneous failures for different subsets of entities. Due to the very large number of subsets and the multiple dependencies among entities, any model that specifies the probability of simultaneous failure for every possible subset of nodes is likely to be complex and of limited usability. Therefore research works have focused on building models that express only the two-way correlations between nodes (e.g., conditional probabilities that one node fails given another node fails) but approximate multiple-way correlations with the two-way correlations. Weatherspoon et al. [90] cluster entities based their two-way correlations and use the clusters as an implicit measure of multiple-way correlations in system design. This heuristic does not produce quantifiable results and, hence, cannot be used to develop systems with a prescribed availability requirements. Other models that approximate multiple-way correlations from two-way correlations are proposed in [5][65]. These models, designed to capture failures of m-out-of-n entities, do not allow for calculating the probability of failure for specific subsets of entities. More importantly, their method provides limited intuition about the nature and the cause of correlated failures and this impairs the accuracy and consistency of entity correlations in the model.

CHAPTER III

STATIC TOPOLOGY CONFIGURATION FOR SERVICE OVERLAY NETWORK

3.1 Introduction

While the Internet has been fully commercialized and evolved into a ubiquitous medium of communication, its native routing infrastructure has become resistant to fundamental changes. This hinders the development of new network functionality (e.g., multicast, QoS) that heavily rely on such fundamental changes. The use of *overlay networks* has been proposed as an alternative solution that can potentially provide the desirable flexibility and control of the routing infrastructure [70, 86, 32, 9]. Researchers have successfully used overlay networks to solve problems in various areas. For example, overlay networks have been employed to implement application layer multicast [20, 7, 68, 59], provide testbeds for new technologies [33, 28, 69], circumvent BGP faults and constraints [2], and provide countermeasures to DoS attacks [48]. In this thesis, we focus our discussion on *service overlay networks* [26, 2, 89] that are deployed and maintained by *overlay network providers*. Overlay network providers deploy a number of specially designed *overlay nodes* across the Internet. As third-party service providers, on one hand, these overlay network providers contract with underlying ISPs and buy network bandwidth between these overlay nodes, and on the other hand, these overlay network providers provide value-added network services to end-systems, who access the overlay networks through one of the overlay nodes. Traffic between end-systems is carried by and routed through the overlay networks instead of the native networks.

One of most important issues in the design of a service overlay network is the configuration of its overlay topology. Positioned between the native networks and the ultimate customers, an overlay topology constructed in favor of both could significantly improve the

performance or reduce the operation cost of the whole system. Consider the four-node example overlay network shown in Fig. 2. Fig. 2.a characterizes the service agreement between the overlay network provider and the underlying ISPs; the labels on the dashed lines show the operation costs of the potential overlay links between every pair of nodes — the price ISPs charge the overlay network provider for shipping one unit of data over these overlay links. Fig. 2.b shows a snapshot of the communication requirements aggregated over all customers; the labels on the lines (or the thickness of the lines) denote the data rates between every pair of overlay nodes, on behalf of the customers. Fig. 2.c shows a candidate overlay topology (Topology A); each edge denotes an overlay link. Provided that the overlay network ships data over the overlay paths that incur the minimum operation cost, the flow of data is shown in Fig. 3.a. Fig. 3.b presents the flow of data if another topology (Topology B) is adopted instead. In terms of operation cost, Topology B is not as good as A — although some data can now take a lower-cost path, a higher volume of data have to take a higher-cost path. Generally, the operation cost of an overlay varies when it runs on different overlay topologies and this raises the problem of finding the overlay topology that minimize the operation cost for given communication requirements.

In this chapter, we study the static topology configuration problem for service overlay networks. The remainder of this chapter proceeds as follows. In Section 3.2, we identify relevant factors in different layers of the system, discuss types of cost that could be incurred in the system, and formally define static topology configuration problem for service overlay networks. In the section, we also give an integer linear programming formulation for static topology configuration problem. Such a formulation allows general tools for solving integer linear programming to be applied to the static overlay topology configuration problem, though their applicability is limited to only small networks due to computational complexity of these tools. In Section 3.3, we discuss the complexity of the static overlay topology configuration problem and show that the problem is NP-hard. We then present a simulated annealing based heuristic method of finding the optimal solutions for the static topology configuration problem. We evaluate our method in Section 3.4 and summarize in Section 3.5.

3.2 Problem Formulation

As shown in Fig. 2, once an overlay topology is established the operation cost for carrying traffic over a service overlay network depends on three factors: the operation cost for shipping data over overlay links, the aggregated communication requirements over all customers and the overlay topology. In this thesis, we assume the overlay network provider is charged by the underlying ISPs an operation cost proportional to the amount of traffic carried on an overlay link once the overlay link is established. We use a *link cost matrix* d to denote the operation cost per unit of data (e.g., bits, bytes) for the overlay link between every pair of nodes. The aggregated communication requirements over all customers are characterized with a matrix of end-to-end traffic rates between every pair of overlay nodes, named *communication patterns*.

The *overlay topology*, is a graph $G = <V, E>$, where V includes all the nodes in the system and E includes all the established overlay links between the nodes. Theoretically, there are a total of $2^{n(n-1)/2}$ possible overlay topologies over n nodes. However, not all of these topologies are desirable in practice. An overlay topology is usually required to be *connected* so that every node remains its contact with the rest of the overlay network. Furthermore, while some research works in overlay networks assume fully-meshed overlay topology, in this thesis we consider more general cases where the overlay topology are *degree-bounded*, i.e., the number of direct neighbors of a node is limited by an upper bound. Service overlay network providers may prefer a degree-bound topology over a fully-meshed one for several reasons. First, underlying ISPs may charge the overlay network providers for a certain amount of fixed maintenance cost for an overlay link in addition to operation cost proportional to the actual bandwidth usage. Second, maintaining an overlay link between two overlay nodes incurs control overhead to the overlay network provider themselves(e.g., link condition probing overhead [2, 58]). Due to these reasons, it can be impractical or undesirable for a overlay network provider to maintain a fully-meshed topology. In this thesis, we consider only overlay topologies that are connected and degree-bounded. We

assume for simplicity that all nodes are subject to the same degree bound K.[1] We denote the set of feasible overlay topologies by 0-1 adjacent matrices $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$.

3.2.1 Static Overlay Topology Design Problem

In an overlay network with topology $\mathcal{T}$, the cost of delivering a unit of data from one node u to another node v is the sum of the operation costs of all overlay links on their overlay routing path. Formally, assume the overlay routing path between u and v is $\mathcal{P}^{u,v}$, then the cost of delivering one unit of data from u to v through the overlay network is:

$$\mathcal{L}^{u,v}(\mathcal{T}) = \sum_{(u,v)\in\mathcal{P}^{u,v}} d(u,v) \tag{1}$$

Then an overall operation cost to support a communication pattern $\mathcal{C}$ on this topology is

$$f(\mathcal{C},\mathcal{T}) = \sum_{u,v\in V} \mathcal{L}^{u,v}(\mathcal{T}) \cdot \mathcal{C}(u,v) \tag{2}$$

in every unit of time. Obviously, function $f(\mathcal{C},\mathcal{T})$ is minimized for a given $\mathcal{C}$ and a given $\mathcal{T}$ if the data are routed through the overlay using minimum-operation-cost paths, which are equivalent to shortest paths if we consider $d(u,v)$ as the distance between u and v. In this thesis, we assume the data are always routed through the overlay network following the minimum-operation-cost paths.

The static overlay topology design problem is the problem of finding an overlay topology $\mathcal{T}$, under the constraints of connectivity and degree-bound, that can minimize the cost function $f(\mathcal{C},\mathcal{T})$ for an communication pattern $\mathcal{C}$. We term such a topology *optimal-static topology* for $\mathcal{C}$ and denote it by $\mathcal{T}^*(\mathcal{C})$. Same as most of topological design problem, the static overlay topology design problem can be modeled as an integer linear programming problem(Section 3.2.2) and is an NP-hard problem(Section 3.3.1).

3.2.2 An Integer Linear Programming Formulation

In this section, we give an integer linear programming formulation for static topology configuration problem of the static topology configuration problem formulated in Section 3.2.1.

[1] Specifying a degree bound for each node does not change the complexity of the problems discussed in this thesis, nor does it affect the applicability of our solutions.

Such a formulation allows general tools for solving integer linear programming to be applied to the static overlay topology configuration problem, though their applicability is limited to only small networks due to the computational complexity of these tools. The static overlay topology configuration problem can be formulated as the following integer linear programming problem:

$$\min \quad \sum_{s,t} w_{st} \sum_{i,j} p_{ij}^{st} l_{ij} \tag{3}$$

subject to

$$\forall\ s,t \quad \sum_{i} p_{is}^{st} + \sum_{j} p_{sj}^{st} = 1 \tag{4}$$

$$\forall\ s,t \quad \sum_{i} p_{it}^{st} + \sum_{j} p_{tj}^{st} = 1 \tag{5}$$

$$\forall\ s,t,k \quad \sum_{i} p_{ks}^{st} + \sum_{j} p_{kj}^{st} = 2f_{k}^{st} \tag{6}$$

$$\forall\ i,j \quad \sum_{s,t} p_{ij}^{st} \leq \frac{1}{2} N(N-1) e_{ij} \tag{7}$$

$$\forall\ k \quad \sum_{i} e_{ik} + \sum_{j} e_{kj} \leq K \tag{8}$$

In the formulation, N denotes the number of nodes, w_{st} denotes the data demand between node s and node t, l_{ij} denote operation cost rate for the potential overlay link between nodes i and node j through the underlying native network. K denotes the degree bound. There are two major types of binary variables:

- Binary variables $e_{ij}, 1 \leq i < j \leq N$ describe the topology of the overlay network. If there is an overlay link between nodes i and j, then e_{ij} takes value 1; otherwise, it takes value 0.

- For any s and t where $1 \leq s < t \leq N$, binary variables $p_{ij}^{st}, 1 \leq i < j \leq N$ describe a path between nodes s and t. If the overlay link between nodes i and j is on the path, p_{ij}^{st} takes value 1; otherwise p_{ij}^{st} takes value 0.

There are three types of constraints.

- Constraints (4-6) make sure the path actually connects nodes s and t. Binary variables f_{k}^{st} are used to model the integer set $\{0, 2\}$.

- Constraint (7) makes sure that every path is a valid path contained in the overlay topology.

- Constraint (8) states the degree bound.

The model is accurate only if $w_{st} > 0$ for all pairs of s and t. In problems where w_{st} could be zero, we can replace w_{st} with a small enough positive number if it is zero. The replacement does not change the optimality of the optimal solution.

3.3 Optimization Techniques

3.3.1 Problem Complexity — Proof of NP-Hardness

In this appendix, we prove the NP-hardness of the static topology configuration problem formulated in Section 3.2.1. To facilitate the proof, we first rephrase the problem to the following graph problem, which we name as *Minimum Communication Cost Spanning Subgraph (MCCSS)* problem.

MINIMUM COMMUNICATION COST SPANNING SUBGRAPH

INSTANCE: Complete graph $G =< V, E >$, length $l(e) \in \mathbf{R}$ for each $e \in E$, weight $w(u, v)$ for each pair of vertices (u,v) from V, and integer $K \geq 2$.

SOLUTION: A connected graph $G' =< V, E' >$ where $E' \subset E$ and the degree of every vertices in the G' is no greater than K.

MEASURE: The weighted summation over all pairs of vertices of the length of the shortest path between the pair in G', i.e., $\sum_{u,v \in V} L(u, v) \cdot w(u, v)$, where $L(u, v)$ denotes the summation of the length of the edges on the shortest path joining u and v in G'.

Obviously, the MCCSS problem is equivalent to the original problem. In the remainder of this section, we prove that the MCCSS is NP-hard. The proof contains two steps. In the first step, we induce the MMCCSS problem to one of its subproblems, named *Minimum Routing Cost Spanning Subgraph(MRCSS)*. In the second step, we induce the MRCSS problem to the Hamiltonian Path problem, which is known to be NP-complete.

First, by setting the weight $w(u, v)$ in the MCCSS problem to 1 for all pairs of vertices from V, we get the MRCSS problem.

MINIMUM ROUTING COST SPANNING SUBGRAPH

INSTANCE: Complete graph $G =< V, E >$, length $l(e) \in \mathbf{R}$ for each $e \in E$, and integer $K \geq 2$.

SOLUTION: A connected graph $G' =< V, E' >$ where $E' \subset E$ and the degree of every vertices in the G' is no greater than K.

MEASURE: The summation over all pairs of vertices of the cost of the shortest path between the pair in G', i.e., $\sum_{u,v \in V} L(u, v)$, where $L(u, v)$ denotes the summation of the length of the edges on the shortest path joining u and v in G'.

Obviously, MRCSS problem is a subproblem of the MCCSS problem. To prove the NP-hardness of the MCCSS problem, we need only to show the NP-hardness of the MCRSS problem. We accomplish this by inducing the MCRSS problem to the Hamiltonian Path problem, which is known to be NP-complete and defined as follows.

HAMILTONIAN PATH

INSTANCE: Graph $G =< V, E >$.

QUESTION: Does G contain a Hamiltonian path?

PROOF: For any instance of the Hamiltonian Path problem, $G =< V, E >$, we can construct a corresponding instance of the MCRSS problem in polynomial time as follows: First, we construct a complete graph $\bar{G} =< V, \bar{E} >$. Second, for any $e \in \bar{E}$, we let $l(e) = 1$ if $e \in E$ and $l(e) = \infty$ if otherwise. Third, we let $K = 2$.

We claim that G contains a Hamiltonian path if and only if the routing cost of the minimum routing cost spanning subgraph (that is connected and subject to degree-bound 2) of $\bar{G}$ is less than ∞. Now we prove the claim. Assume the minimum routing spanning subgraph of $\bar{G}$ is $\bar{G}'$. First, for any Hamiltonian path of G, denoted with H, $G^H =< V, H >$

is a feasible (i.e., connected and conforming to degree-bound 2) spanning subgraph of $\bar{G}$. Note that G^H contains only the edges whose $l(e) = 1$ in $\bar{G}$, so the routing cost of G^H must be less than ∞. Because the routing cost of G^H is no less than that of $\bar{G}'$, the routing cost of $\bar{G}'$ must be less than ∞. Conversely, if the routing cost of $\bar{G}'$ is less then ∞, $\bar{G}'$ must contains only the edges from G. Since $\bar{G}'$ is connected and conforms to degree-bound 2, the edges from $\bar{G}'$ must form a Hamiltonian path or a Hamiltonian circuit of G. In either case, G contains a Hamiltonian path.

The above claim allows us to convert an instance of the Hamiltonian Path problem to an instance of MCRSS problem and thus the MCRSS problem is induced to the Hamiltonian Path problem. Since the latter is NP-complete, the former is NP-hard. Since the MCRSS problem is a subproblem of the MCCSS problem, the latter is NP-hard too.

3.3.2 A Heuristic Approach Based On Simulated Annealing

In this section, we develop solutions to static overlay topology configuration problem, the problem of finding the static-optimal topology for a given single communication pattern. The problem is previously formulated in Section 3.2.1 and is proved to be NP-hard in Section 3.3.1. A heuristic algorithm is called for.

Static topological design problems have been extensively studied for the construction of native networks. The class of problems have many formulations depending on specific design objectives and scenarios, and most of them are NP-Hard problems. An excellent compilation of problems and solutions can be found in [72]. Study of static topological design problems is still rare in the context of service overlay networks. An overlay topology design problem presented in [89] share certain similarity with our formulation, but the two problems are fundamentally different: the complexity of our problem is caused by the degree bound while that of the other problem is caused by overlay entrance selection. In the remainder of this section, we present a *Simulated Annealing* based algorithm for our problem. The application of some basic optimization techniques we use, such as simulated annealing and topology perturbation, has root in the literature of topological design for native networks (e.g., [82, 11]).

PROCEDURE FIND-OPTIMAL-TOPOLOGY
Initialize :
construct a random initial point and calculate its cost;
— comment: a random ring topology
let *Optimal-Topology* be the current topology; let *Optimal-Cost* be the current cost;
estimate the initial temperature T_0 and let current temperature $T = T_0$;
Repeat phases until T is close enough to zero:
set the stop temperature T_e of this phase to $\frac{1}{\phi}T$;
— comment: $\phi > 1$
Repeat moves until T reaches T_e:
randomly mutate the current topology using one of the operations;
redo the above step until the resulting topology is a connected one ;
evaluate cost function f of the new topology, calculate the change, Δf;
if $\Delta f < 0$, accept the new topology; else accept with probability $e^{\frac{\Delta f}{T}}$;
if $\Delta f < 0$, let this topology the new optimal one.
— comment: update*Optimal-Topology* and *Optimal-Cost*;
let $T = \rho T$;
— comment: geometric cooling; ρ is typically in the range of [0.95,1)
let $T = \varsigma T_e$;
— comment: reheating; $\phi > \varsigma > 1$
Output *Optimal-Topology* and *Optimal-Cost*;

Figure 7: Pseudo-code of Finding Optimal-Static Topology Using Simulated Annealing

Simulated Annealing is a global optimization meta-algorithm based on an analogy with the physical process of annealing [51]. Its general structure has been applied to many combinatorial problems with good results [45]. Starting from an initial solution and an initial *temperature*, the meta-algorithm walks randomly in the solution space. Cost-decreasing moves are certainly accepted while cost-increasing are accepted only with a probability $P = e^{\frac{\Delta f}{T}}$, where Δf is the increase in the cost function f and T is the temperature. By decrementing and possibly incrementing the temperature following a deliberate *annealing schedule*, this probabilistic process will finally stabilize at a final solution. The overall structure of our algorithm is shown in Fig. 7.

Initial Point The algorithm starts from a random overlay topology that is connected and subject to degree bound K.

Searching for a Solution Finding another feasible solution in the neighborhood of the current feasible solution is the key task in the probabilistic walk-around. We accomplish the task by first mutating the current feasible overlay topology into a *slightly different* topology

that still satisfies the degree bound and then verifying the connectivity of the new topology. We use the following mutating operations in the algorithm.

- Operation 1: This operation randomly chooses two nodes A and B whose degrees are less than K and adds an overlay link between the two nodes.
- Operation 2: This operation randomly chooses four nodes that form a local setup like Fig. 8.a and converts it to either Fig. 8.b or Fig. 8.c.

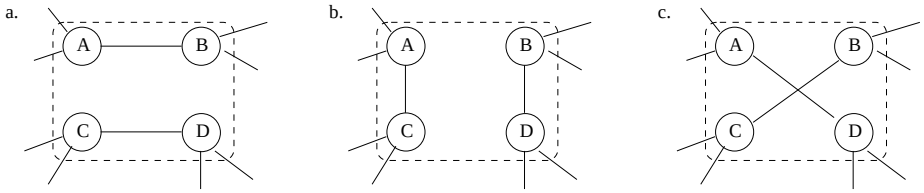


Figure 8: Mutating Operation 2

- Operation 3: This is a derivative of Operation 2. This operation randomly chooses four nodes that form a local setup like Fig. 9.a and converts it to either Fig. 9.b or Fig. 9.c.

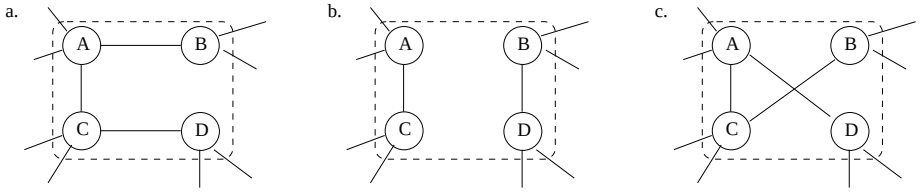
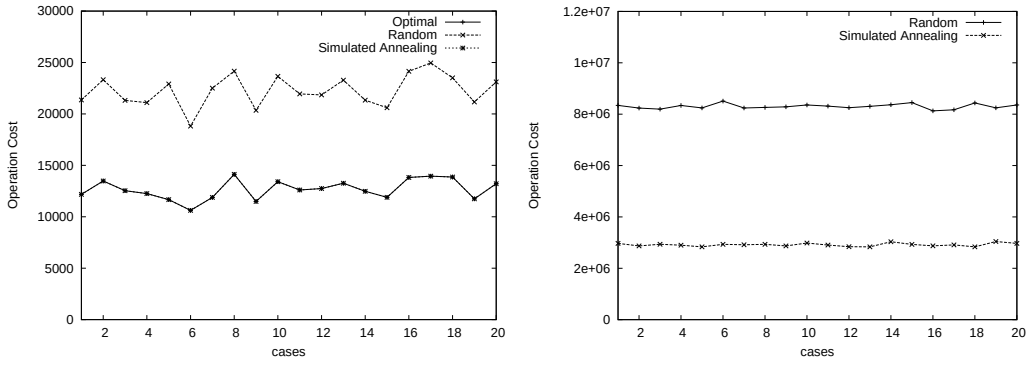


Figure 9: Mutating Operation 3

These operations allow the algorithm to move towards any feasible solution gradually and continuously without ever leaving the space of feasible solutions.

Initial Temperature Kirkpatrick [50] suggested to set the initial temperature to one that results in an average acceptance probability of about 0.8 for uphill moves from the initial point. We estimate the initial temperature T_0 in the following way: we attempt a number of random cost-increasing moves, all from the initial point, observe the average increase in cost, $\overline{\Delta f}$, and then calculate the initial temperature T_0 by: $T_0 = \frac{\overline{\Delta f}}{\ln(0.8)}$.



(a) Small Problems (7 nodes, degree=2)

(b) Large Problems (100 nodes, degree=4)

Figure 10: Performance of Subroutine for Finding Static-Optimal Topology. Note that in this experiment the "Optimal" line and the "Simulated Annealing" line in (a) exactly overlap.

Annealing Schedule In the literature, the choice of annealing schedule is quite problem specific. In several large combinatorial problems, researchers use geometric cooling for expediency yet get good result with the help of reheating [1]. We follow the same direction. We try different types of schedules and find the following one is good for our problem. The schedule is composed of phases. In the pth phase, starting from the initial value T_0^p, the temperature is multiplied by ρ each time the algorithm attempts a move (no matter if the move is accepted or rejected), i.e., $T_{k+1}^p = \rho T_k^p$, where ρ is typical a value between 0.95 and 1. When the temperature reaches down $\frac{1}{\phi}T_0^p$, where $\phi > 1$, the algorithm ends the current phase. It reheats the system by multiplying the temperature by ς, where $\varsigma > 1$, and enters the $(p+1)$th phase with a starting temperature $T_0^{p+1} = \frac{\varsigma}{\phi}T_0^p$. By choosing $\phi > \varsigma$, e.g., $\phi = 3$ and $\varsigma = 2$, the temperature goes down after each phase and eventually gets close to zero, where the annealing schedule ends and the algorithm exits.

3.4 Performance

In this section, we evaluate the heuristic approach presented in Section 3.2. When the problem is small (e.g., the number of node is less than 10), we compare the result of the subroutine to the real static-optimal topology and randomly generated topologies. When the problem is large, we are only able to compare the result to the randomly generated ones.

We experiment on randomly generated communication patterns using a randomly generated matric of overlay link operation costs. Fig. 10.a shows the experimental result for 20 randomly generated cases with 7 nodes and degree bound 2. The setup of the case problems is as follows: we generated a random underlying network by setting the distance between pairs of node to a random value in range $[10, 100]$. We then generate 20 communication patterns, within each of which we set the data demand rate between every pairs of node to a random value in range $[0, 10]$. For each of the 20 cases, the figure shows the operation cost of the topology produced by our algorithm, the real optimal one and the random one, respectively. For the simulated annealing, we set $\rho = 0.99$, $\phi = 3$ and $\varsigma = 2$ in the annealing schedule. The cost for the random topology is the average over 30 randomly generated feasible topologies. The figure shows that for all these 20 cases, our algorithm finds the exact optimal solution. The figure also shows that overlay topologies produced by our algorithm perform much better than random ones. Fig. 10 shows results for a problem with 100 nodes with a degree bound of 4. Other experiment parameters are the same as in the previous set of experiments. The figure reaffirms that our algorithm produces topologies with improved performance.

3.5 Summary

In this chapter, we have studied the static topology configuration problem for service overlay networks. We have identified relevant factors in different layers of a service overlay network, discussed types of cost that could be incurred in the system, and formally defined the static topology configuration problem for service overlay networks. We have given an integer linear programming formulation for the problem. Such an ILP formulation allows general tools for solving integer linear programming to be applied to the static overlay topology configuration problem, especially for the overlay networks that have only a small number of nodes. We have then discussed the complexity of the static overlay topology configuration problem and shown that the problem is NP-hard. We have presented a simulated annealing based heuristic method of finding the optimal solution for the static topology configuration problem. Our evaluation has shown that the heuristic method has good performance and

is applicable to service overlay networks with a large number of nodes.

CHAPTER IV

DYNAMIC TOPOLOGY CONFIGURATION FOR SERVICE OVERLAY NETWORK

4.1 Introduction

In a service overlay network described in Section 3.1, if the communication requirement is constant over time, the optimal choice of overlay topology is static. If communication requirements change over times, for example, from the ones shown in Fig. 2.b to the ones shown in Fig. 3.c, the overlay topology may need to be reconfigured. While the static topological design problems have been extensively studied for native networks, the design of dynamic topology did not get the same amount of attention because the hard-wired topologies of native networks are usually not reconfigurable in small time scale. In overlay networks, however, the topology reconfigurability is one of their major strength and appeal and the dynamic topology design problems become interesting. For overlay networks with tens of overlay nodes, it is feasible for the providers to monitor and statistically model the communication requirements in the system. The topology reconfigurability allows an overlay network to be tuned dynamically when the communication requirements change [85, 87].

A question needs be answered, however, namely, *when and how the overlay topology should be reconfigured under dynamic communication requirements.* One may suggest that whenever the communication requirements change, the topology of the overlay network should be immediately reconfigured to the topology that minimizes the operation cost for the new communication requirements (and the dynamic overlay design problem is reduce to the static one). However, while overlay topology is reconfigurable in small time scale, changing overlay topology is not cost-free; it may incur both management overhead as well as potential disruption of end-to-end flows: overlay links need to be established or torn down; routing tables need to be updated; data in transit may get lost, delayed, or

erroneously routed. In the presence of these costs, changing the overlay topology for every change in communication requirements may not be the best policy. Intuitively, if in the long run the benefits of making a change in the overlay topology cannot justify the costs of the change, the overlay topology should not be changed. Even if a change is favorable, the next overlay topology should take into account the long-term changes in communication requirements.

In this chapter, we formulate the dynamic overlay topology configuration problem, i.e., the problem of determining dynamic topology reconfiguration in the presence of dynamic communication requirements. Particularly, we concentrate on reconfiguration policies that guide the topology selection any time the communication requirements change. Similar problems have been previously studied for the design of optical networks [80, 6] but have not been systematically studied in the context of service overlay networks. To solve the dynamic overlay topology configuration problem, we systematically observe how optimal reconfiguration policies is affected by the reconfiguration overhead, through case studies on small systems. Based on the observation, we propose heuristic methods for constructing different flavors of reconfiguration policies for large systems. In the process of solving this problem, much of the notation and formulation presented in Chapter III is still used in this chapter, and the solution previously developed for the static overlay topology design problem in Chapter III is used as a subroutine in the solutions to the dynamic problems in this chapter. Our work does not only provide solutions to practical overlay topology design problems, but also provides theoretical evidence for the advantage of overlay network due to its configurability.

The remainder of this chapter proceeds as follows. In Section 4.2, we discuss types of cost incurred in the system during an overlay topology reconfiguration process, and formally define the decision making problem in dynamic topology reconfiguration. In Section 4.3, we observe the structure of the problem, as well as the properties and structures of the optimal reconfiguration policies, through experiments on small Markovian systems. In Section 4.4, we use these observations as heuristics and propose methods for constructing good reconfiguration policies that approximate the optimal ones for large systems. We

evaluate our methods in Section 4.5 and conclude in Section 4.6.

4.2 Problem Formulation

For a service overlay network where the communication requirements change over times, we denote the communication requirements at time t by $X(t)$ and assume $X(t)$ takes on values from a set of distinct communication patterns, $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s\}$. An *overlay topology reconfiguration policy* is the sequence of overlay topologies used by an overlay network over time, denoted by $Y(t)$. The problem we are addressing in this chapter is finding the optimal reconfiguration policies $Y(t)$ for a given communication requirements $X(t)$. This determination is guided by the cost functions, including not only the static operation costs already discussed in Chapter III, but also reconfiguration costs which we discuss later in this chapter. If the communication requirements $X(t)$ is constant over time, i.e., $X(t) = \mathcal{C}$ for any t, then the optimal overlay topology reconfiguration policy is one that always uses the optimal-static topology for $\mathcal{C}$, i.e., $Y(t) = \mathcal{T}^*(\mathcal{C})$ for any t. The problem formulation in Section 3.2.1 still applies when the communication requirements do not change over time. In this chapter, however, we are interested in the more general cases where the communication requirements change over time.

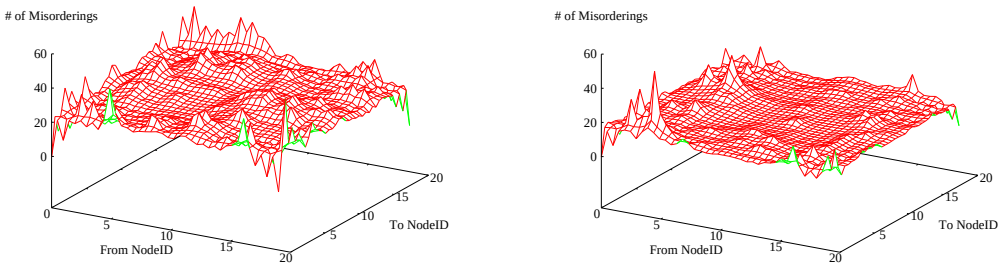
4.2.1 Dynamic Overlay Topology Design Problem

We consider two categories of costs in an overlay network: *Occupancy Cost* and *Reconfiguration Cost*. The occupancy cost is incurred while the overlay network is configured in a particular topology while the reconfiguration cost is incurred whenever the overlay topology is reconfigured.

Occupancy Cost The occupancy cost is the total operation cost for the overlay network to deliver the traffic specified by the dynamic communication requirements $X(t)$ over the dynamic overlay topology $Y(t)$ specified by the overlay topology reconfiguration policy. That is

$$COST_o(\Delta t) = \int_0^{\Delta t} f(X(t), Y(t))dt \tag{9}$$

where Δt is the time horizon of interest and the function $f(\mathcal{C}, \mathcal{T})$ is defined in Eq. 2.



(a) Protocol A (b) Protocol B

Figure 11: Flow Disturbance Caused by Topology Reconfiguration

Reconfiguration Cost Every time the system reconfigures its overlay topology to adapt to changes in communication requirements, a reconfiguration cost is incurred. This cost is the overhead or the impairment to performance incurred by the transition from one overlay topology to another.

Various costs could be incurred during a topology reconfiguration, depending on the implementation details of the overlay. For example, establishing and tearing down overlay links incur *control and management overhead*, especially when underlying ISPs are involved in the processes. Furthermore, data in transit during topology reconfiguration is subject to routing disturbance that incurs *rerouting overhead*. Depending on the overlay implementation, when overlay topologies and routing tables change, data in transit could be simply dropped by intermediate nodes and requires end-to-end retransmission, or be rerouted, maybe several times, wandering through a path with a high operation cost. Finally, rerouting overhead can be magnified at the end-systems. Data loss and misordering caused by overlay topology reconfiguration are much more significant than those caused by factors in underlying networks. They last longer and their impact is system wide. End-systems' flow control mechanisms that assume low loss rate and short misordering sequence, e.g., TCP's window control system, the impact of overlay topology reconfiguration can be very interruptive.

Figure.11 gives evidence to the existence of reconfiguration cost, e.g., data misordering. The experiments are conducted over the PlanetLab [69] using 20 *.edu* nodes in North

America, which form a simple overlay. Each node runs a UDP-based routing daemon and is remotely controlled from a monitor in Georgia Tech using XML-RPC. To conduct the experiment, we let each node generate one packets per 10 millisecond to every other nodes and count the pairs of misordering packets caused by an overlay topology reconfiguration. To reconfigure the topology, the monitor sends out XML-RPC calls simultaneously to every nodes, commanding them to update their neighboring status and routing tables, using one of two protocols. In Protocol A, each nodes updates its routing table immediately after receiving the XML-RPC calls. Due to variation in their network distance from the monitor, the routing tables are actually updated at different time at different nodes. In Protocol B, the monitor provides a future time point in the XML-RPC calls and ask all nodes to clock-synchronize their updating at that time point. The figures show the resulting amount of misordering for each end-to-end flow when the overlay makes transition from one degree-4 topology to another one. The figures show that Protocol A incurs less flow disturbance than B. But in both protocols, the disturbance on flows is significant and system-wide.

The formulation of the reconfiguration cost in an topology transition naturally varies for different implementation of overlay network (e.g., protocol for the transition), and it is not goal to deal with each specific implementation in this thesis. Instead, at this stage, we are most interested in understanding the dynamic overlay topology design problem in a more general manner. In this thesis, we use the total number of overlay links that need to be changed during a overlay transition as a general approximate metric for the reconfiguration cost, with a specific scaling factor for each specific system. Formally, the metric is

$$\mathrm{g}(\mathcal{C}_{old}, \mathcal{T}_{old}, \mathcal{C}_{new}, \mathcal{T}_{new}) = \sum_{u,v} |\mathcal{T}_{old}(u,v) - \mathcal{T}_{new}(u,v)| \tag{10}$$

where $\mathcal{C}_{old}, \mathcal{C}_{new}, \mathcal{T}_{old}, \mathcal{T}_{new}$ are the old and new communication patterns and overlay topology respectively.[1] The total reconfiguration cost over a time horizon Δt, denoted by $COST_r(\Delta t)$, is the sum of the reconfiguration costs incurred for all overlay topology transitions that happen during Δt. The reconfiguration policy construction methods developed in this thesis allows plugging in other formulations of function $f(\cdot)$, and the use of other

[1]The system specific weighting factor is absorbed in β in Eq.sessioncost.

formulations does not conflict in intuition with heuristics used by these methods. In Section 4.5.2, we test the use of some other formulations in our policy construction methods and find that the goodness of resulting reconfiguration policies remains.

Overall Cost The overall cost of using the overlay over a time horizon Δt is the sum of both the occupancy cost and the reconfiguration cost incurred in the period. Formally,

$$COST(\Delta t) = COST_o(\Delta t) + \beta \cdot COST_r(\Delta t) \tag{11}$$

where factor $\beta \in [0, 1]$ reflects the relative weight of reconfiguration cost and occupancy cost; its actual value depends on the implementation details in how the topology transitions are conducted and how different performance/cost factors (e.g., data loss rate, retransmission, performance of end-systems) influence and are evaluated against each other. The long-run average of the overall cost is

$$\liminf_{\Delta t \to \infty} \frac{COST(\Delta t)}{\Delta t} \tag{12}$$

4.2.2 Topology Reconfiguration Policies

Topology Reconfiguration Policy Given the communication patterns, $X(t)$, a *reconfiguration policy*, $Y(t)$, is essentially a set of rules specifying when and how the overlay topology should be reconfigured. The following are three examples.

- Policy 1: Whenever the communication pattern changes, the overlay topology should be reconfigured to a random one.
- Policy 2: Every 10 minutes, the overlay topology should be reconfigured to the static-optimal one for the current communication pattern.
- Policy 3: Whenever the communication pattern changes, the overlay topology should be reconfigured to static-optimal one for the new communication pattern.

In this thesis, we concentrate on the class of reconfiguration policies with the following properties:

- Memoryless. Assume the overlay topology changes at time t, in a memoryless reconfiguration policy, the choice of the new overlay topology $Y(t+\delta)$ depends only on the current communication pattern $X(t)$, the current overlay topology $Y(t)$ and the new communication pattern $X(t+\delta)$. All the three example policies have this property. Compared with reconfiguration policies that have to keep the history of previous decisions, memoryless policies are easier to design, analyze and implement. In fact, for many reconfiguration policies that keep a limited history of previous decisions, it is possible to convert them into memoryless ones by, i.e., combining a sequence of history states into a single (though more complex) state.

- Reactive. In a reactive reconfiguration policy, a change in overlay topology can only be triggered by a change in the communication pattern. Policy 1 and 3 have this property while the Policy 2 does not. Again, compared with reconfiguration policies with arbitrary timing in reconfiguration, reactive policies are easier to design, analyze and implement. In fact, if the dynamics of communication pattern is memoryless, a non-reactive reconfiguration policy can always be converted to a reactive one that incurs no more overall cost, by, for example, moving up the timing of a reconfiguration to the time when the last change in communication pattern happens or postpone it to the time when the next change in communication pattern happens.

- Deterministic. Given $X(t)$, $Y(t)$ and $X(t+\delta)$, in a deterministic reconfiguration policy, the choice of $Y(t+\delta)$ is deterministic rather than probabilistic. Policy 2 and 3 have this property while Policy 1 does not. The space of probabilistic policies is a superset of that of deterministic policies, but theoretically there are unlimited number of probabilistic policies. Popular policy optimization approaches, such as Markov decision process and Howard's policy iteration method do not apply. Furthermore, compared with the deterministic ones, probabilistic policies require extra coordination among distributed overlay nodes so that their probabilistic reconfiguration behaviors are consistent across the whole system.

A reconfiguration policy in this class is essential a function that maps each possible triple of current communication pattern $\mathcal{C}_{old}$, current overlay topology $\mathcal{T}_{old}$ and new communication pattern $\mathcal{C}_{new}$ to a new topology $\mathcal{T}_{new}$. It is possible that $\mathcal{T}_{new} = \mathcal{T}_{old}$, in which case the overlay topology remains the same despite a change in communication pattern. The ideal goal is to find the *optimal reconfiguration policy* that can minimize Eq.12 for an given model of $X(t)$. The problem is NP-hard because even the static version of the problem is NP-Hard. Please refer to Section 3.2 and Section 3.3.1 for a detailed discussion.

4.2.3 Methodology and An MDP Formulation

Our methodology is, therefore, to start by observing the general properties and structures of the optimal reconfiguration policy on small, solvable cases, and then use the observations as heuristics to solve the problem for large cases. Particularly, in Section 4.3, we study systems that have small number of nodes and in which $X(t)$ is a continuous Markov process, intending to understand, for example, how the optimal policies are affected by various characteristics of $X(t)$ and the level of reconfigure cost. Then in Section 4.4, we use our observations as heuristics and propose heuristic-based solutions that are applicable to problems with large number of nodes or with non-Markovian $X(t)$.

Note that for systems where $X(t)$ is a continuous Markov decision process, the dynamic topology design problem can be modeled as a continuous time *Markov decision process* [40]. Each state in the decision process consists of one communication pattern from $X(t)$ and an feasible overlay topology. For example, for the 3-state Markovian model of $X(t)$ show in Fig. 12, if there are only two feasible overlay topologies $\mathcal{T}_1$ and $\mathcal{T}_2$, the corresponding Markov decision process will have six states in total, $\{<\mathcal{C}_i, \mathcal{T}_j> \ | i = 1, 2, 3; j = 1, 2\}$. A policy of the Markov decision process, corresponding to a topology reconfiguration policy in the original problem, is composed of a set of decisions, one at each state. For example, Fig. 13 shows the four alternatives that the decision process needs to decide on when it is in state $<\mathcal{C}_1, \mathcal{T}_1>$. If each of the 6 states has 4 alternatives, then there will be 4^6 possible policies. When the number of nodes in the system is small (e.g., no more than 7) so that the number of states is still manageable, the decision process can be practically solved using

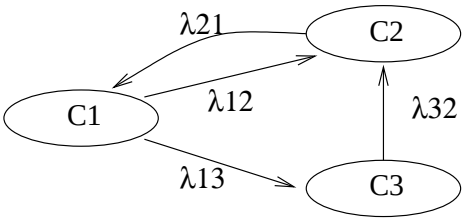


Figure 12: $X(t)$ is a continuous Markov process

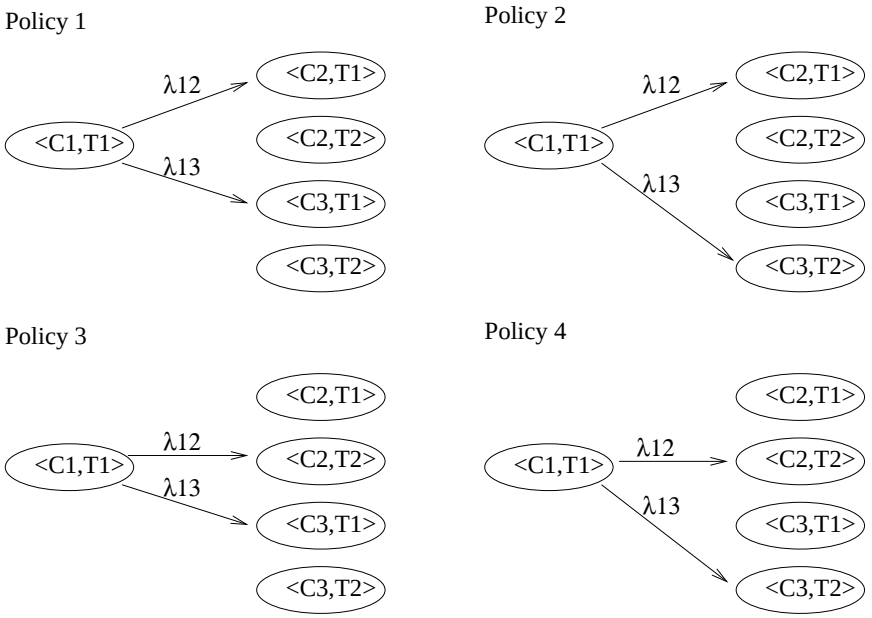


Figure 13: Various Policies Make Different Decision at State $<\mathcal{C}_1, \mathcal{T}_1>$

Howard's policy iteration algorithm [40].

Also note that during the calculation of function $f(\cdot)$ in Eq. 9, we assume the underlying network operation cost matrix is static. This is reasonable in most scenarios because user requirements are usually much more dynamic than underlying network operation cost factors and the latter can therefore be assumed static. It is possible, however, to extend the formulation to incorporate the cases where underlying network operation cost matrix is dynamic. In such cases, the dynamics of underlying network operation cost can be modeled together with the dynamics of users' requirements. The methodology used in this thesis (e.g., MDP model and case studies) still applies in this extended context, and the estimation models for reconfiguration cost still apply. However, the benefit of making a

topology reconfiguration needs a new formulation — it depends on not only the changes in the communication pattern but also the changes in the operation cost matrix. As to the applicability of the heuristics presented in this thesis under this extended model, more investigation is required in the future.

4.3 Properties and Structure of Optimal Reconfiguration Policies

Our goal in this section is to determine the structures and properties of the overlay reconfiguration policies. To that end, we experiment on a sample network overlay with 5 nodes. We set $d(u_i, u_j) = 1$ for any pair of nodes u_i and u_j, so there is not difference between the five nodes in terms of routing in the underlying network. We set the degree bound of the overlay networks to 2, so there are 12 feasible overlay topologies. [2] Fig. 14 shows a simple communication requirement transition model with two communication patterns $\mathcal{C}_1$ and $\mathcal{C}_2$.

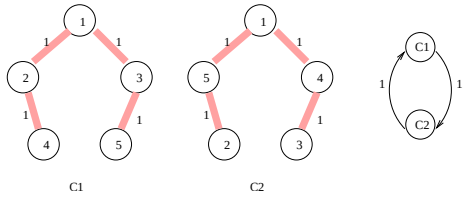
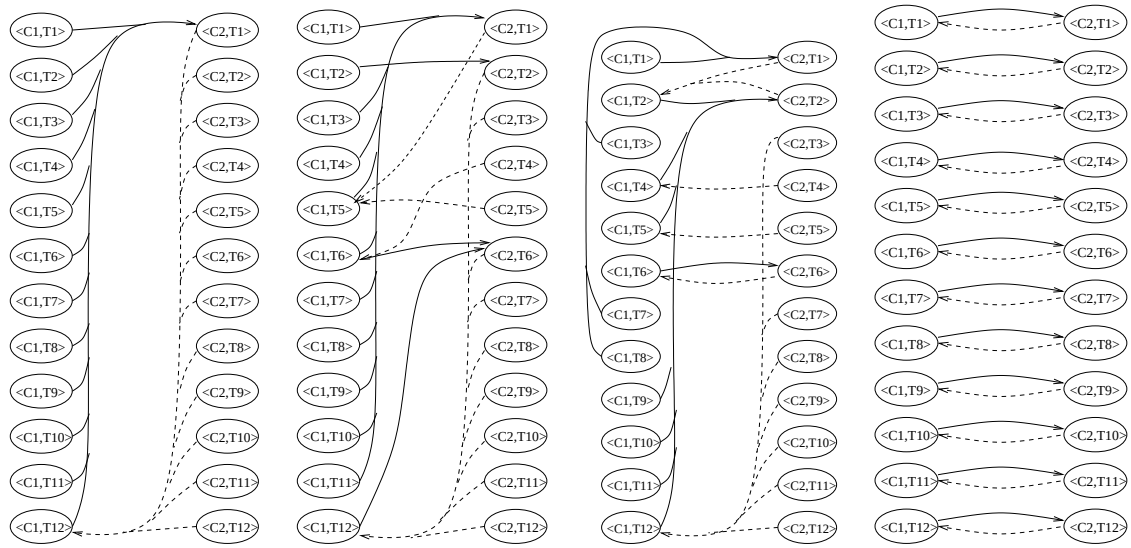


Figure 14: Communication Requirement Transition Model With Two Communication Patterns

4.3.1 Trends of Aggressiveness

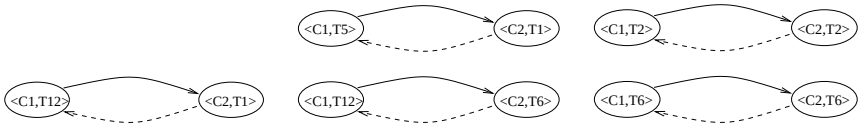
Chains in Optimal Policies Fig. 15.(a) shows the optimal policy when we set the weight of reconfiguration cost(β in Eq. 11) to 0 (which means the reconfiguration cost is not significant relative to the occupancy cost.) Our first observation is that the system is very aggressive. No matter what the old overlay topology is, the optimal policy always chooses overlay topology $\mathcal{T}_1$ or $\mathcal{T}_2$ whenever the communication pattern changes to $\mathcal{C}_1$ or $\mathcal{C}_2$, respectively. Our calculation also shows that $\mathcal{T}_1$ and $\mathcal{T}_{12}$ are the minimum-occupancy-cost topologies for $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Our second observation is that the system, starting

[2] There are actually 72 overlay topologies that are connected and with degree bound 2. For simplicity of presentation, we only count the topologies in which each node's degree is *exactly* 2.



(a) $\beta = 0$ (b) $\beta = 0.3$ (c) $\beta = 0.4$ (d) $\beta = 10000000$

Figure 15: Optimal Policies. Solid lines and dashed lines show how the overlay topology is reconfigured when the communication pattern changes from $\mathcal{C}_1$ to $\mathcal{C}_2$ and from $\mathcal{C}_2$ to $\mathcal{C}_1$, respectively.



(a) $\beta = 0$ (b) $\beta = 0.3$ (c) $\beta = 0.4$

Figure 16: Optimal Reconfiguration Chains. Solid lines and dashed lines have the same meanings as in Fig. 15.

from any state, steers itself into a recurrent chain of states, possibly after a few steps of initial topology reconfiguration, and stays within the chain. We term a chain an *optimal reconfiguration chain* of the optimal policy. Not all feasible overlay topologies show up in the optimal reconfiguration chain. In this example, the chain is composed of states $<\mathcal{C}_1, \mathcal{T}_{12}>$ and $<\mathcal{C}_2, \mathcal{T}_1>$ and contains only two distinct overlay topologies, $\mathcal{T}_1$ and $\mathcal{T}_{12}$.

Effect of Weight of Reconfiguration Cost The system keeps the same optimal policy when we gradually increase the value of β, until it reaches a certain threshold. Fig. 15.(b) presents the optimal policy when β equals 0.3. The optimal policy now has two optimal reconfiguration chains (shown in Fig. 16.(b)) instead of one. Compare to the case when $\beta = 0$, the system is still aggressive; it always reconfigures the overlay topology whenever the communication pattern changes. The choice of optimal reconfiguration chain, however,

now depends on the initial state of the system; the system has become more sensitive to the reconfiguration cost and therefore prefers the "nearest" optimal reconfiguration chain, the one that it can reach with minimum initial reconfiguration cost. Our calculation shows that, however, since the value of β is low, the initial reconfiguration cost varies little when the system starts from various initial states.

We continue to increase the value of β gradually. The system changes its strategy again when the value of β reaches another threshold. Fig. 15.(c) shows the optimal policy when $\beta = 0.4$. Again, the optimal policy has two optimal reconfiguration chains, as shown in Fig. 16.(c). Each optimal reconfiguration chain, however, contains only one overlay topology now,$\mathcal{T}_2$ or $\mathcal{T}_6$. The system becomes even more conservative due to the increased value of β and tries to avoid any reconfiguration except for initial reconfiguration cost for the system to enter an optimal reconfiguration chain. Once the system enters the optimal reconfiguration chains, it will stick with the same overlay topology.

Fig. 15.(d) shows the optimal policy when we raise the value of β to some extreme value. e.g., 10000000. The system becomes most conservative and avoids any topology reconfiguration — the system will stick with its initial overlay topology and totally disregards the occupancy cost. The cost of even a single reconfiguration prohibits any attempt of adjusting to a better overlay topology. Obviously, the optimal policy has 12 optimal reconfiguration chains. (We omit the figure since it is the same as Fig. 15.(d).) The overall cost contains only occupancy cost and varies significantly when the system starts from different overlay topologies.

Effect of Transition Rates We also study the effect of transition rates on optimal policy. We scale the transition rates between communication patterns in Fig. 14 up and down by multiplying all the rates with a common factor γ and observe how the optimal policies are affected. The plots in Fig. 17 present how the number of distinct overlay topologies in the optimal reconfiguration chains [3] changes when we increases the weight of the reconfiguration cost (β); the three plots are for the cases where the scale factor γ is set to 0.01, 1 and 10,

[3]If there are multiple optimal reconfiguration chains in the optimal policy, we take the expected value, assuming the system starts from a random state with equal probability for all states.

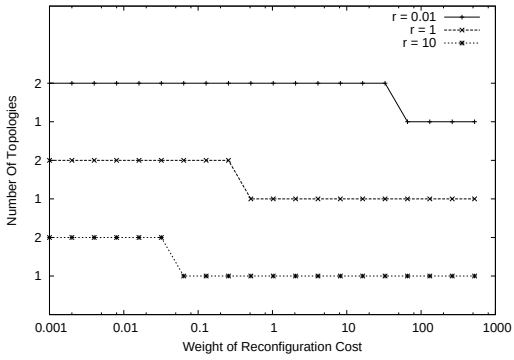


Figure 17: Affect of Transition Rates

respectively. The figure shows that in each plot, there is a turning point where the number of distinctive overlay topologies in the optimal reconfiguration chains turns from 2 to 1, which indicates the system's crossing from being conservative to being aggressive. The figure shows that the higher the transition rates are, the lower the turning point is. The more actively the system changes its communication patterns, the more conservative it is in reconfiguring overlay topology. The system avoids incurring the reconfiguration cost for too many times.

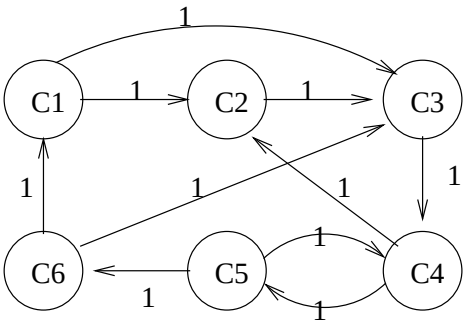
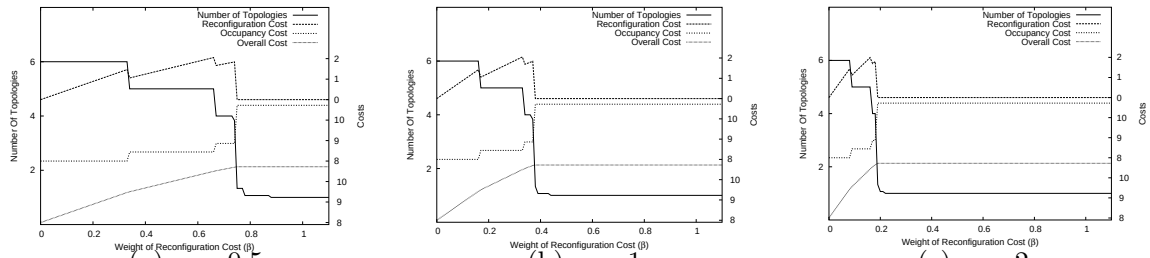


Figure 18: Communication Requirement Transition Model with 6 Communication Patterns

Trends of Aggressiveness and Costs The trends of system's aggressiveness versus the weight of reconfiguration cost and the level of transition rates between communication patterns become even clearer when we experiment with communication requirement transition models with more communication patterns. Fig. 18 presents a communication requirement transition model with six communication patterns. Fig. 19.a plots the number of distinct



(a) $\gamma = 0.5$ (b) $\gamma = 1$ (c) $\gamma = 2$

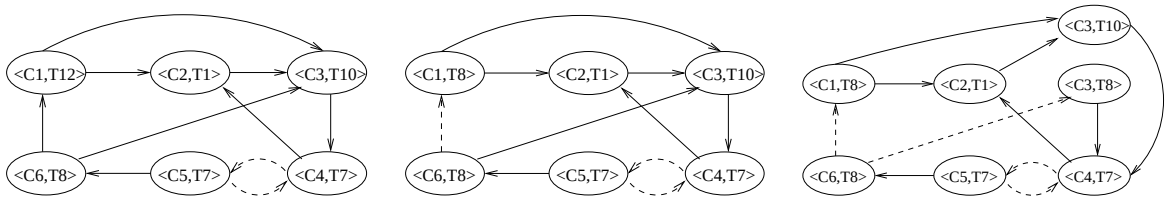
Figure 19: Trends of Aggressiveness and Costs

overlay topologies in the optimal reconfiguration chains, the reconfiguration cost (after being scaled with β), the occupancy cost and the overall cost, respectively.[4] The figure shows that the system's sensitivity to the weight of transition cost is discrete and threshold-based: the whole range of β is divided into ranges by some thresholds; the system's strategy stays basically the same when the value of β stays within the same range but changes significantly when the value of β crosses into another range. On one hand, within the same range, the system's aggressiveness remains in the same level, the occupancy cost remains in the same level, the reconfiguration cost (before weighting with β) remains the same. On the other hand, when the value of β crosses up into another range, the system's aggressiveness drops into a lower level, the occupancy cost jumps up to a higher level, and the reconfiguration cost drops to a new level. The plot for the overall cost, however, is continuous and monotonic. The thresholds and the division of β ranges depend on the level of transition rates in the communication requirement transition model. Fig. 19.(b) and (c) presents the graphs when the transition rates are at two other levels. They show that the β thresholds move to the left when the transition rates are scaled up; the higher the transition rates are, the less aggressive the system is.

4.3.2 Internal Structures of Optimal Reconfiguration Chains

The optimal reconfiguration chains also have internal structures. Fig. 20 presents some optimal reconfiguration chains for the communication requirement transition model shown

[4] Again, this is the expected value, assuming the system starts from a random state with equal probability for all states.



(a)$\gamma = 0.5, \beta = 0.4$ (b)$\gamma = 0.5, \beta = 0.7$ (c) $\gamma = 0.5, \beta = 0.73$

Figure 20: Optimal Reconfiguration Chains. Solid lines represent transitions between states with different overlay topology, while dashed lines represent transitions between states with the same overlay topology.

in Fig. 18. Fig. 20.a presents the optimal reconfiguration chain when $\gamma = 0.5, \beta = 0.4$. It shows that $\mathcal{C}_4$ and $\mathcal{C}_5$ share the same overlay topology $\mathcal{T}_7$ and therefore there is no topology reconfiguration when the system make transitions between $\mathcal{C}_4$ and $\mathcal{C}_5$. Fig. 20.b presents the optimal reconfiguration chain when $\gamma = 0.5, \beta = 0.7$. In addition to $\mathcal{C}_4$ and $\mathcal{C}_5$ share $\mathcal{T}_7$, $\mathcal{C}_6$ and $\mathcal{C}_1$ are also sharing the same overlay topology $\mathcal{T}_8$ now. Informally, we refer to the set of all communication patterns that share the same overlay topology in the optimal reconfiguration chain as a *cluster*. When each communication is associated with only one overlay topology in the optimal reconfiguration chain, the clusters are exclusive of each other, as those in Fig. 20.a and b. In such cases, the formation of clusters in an optimal reconfiguration chain can exactly define the chain: the system keeps its overlay topology when making a transition between communication patterns in the same cluster and reconfigures when it crosses clusters; when a reconfiguration occurs, the system always reconfigures to the overlay topology shared by the next cluster. There are cases, however, that a communication may be associated with multiple overlay topologies in the optimal reconfiguration chain. Fig. 20.c shows an example in which $\mathcal{C}_3$ belongs to two clusters, one including only itself, another one including also $\mathcal{C}_1, \mathcal{C}_6$. When clusters overlap, the formation of clusters in an optimal reconfiguration chain cannot exactly define the chain. Our experiment, though, shows that most communication patterns are associated with only one overlay topology in optimal reconfiguration chains.

We observe that the formation of clusters within an optimal reconfiguration chain is affected by two categories of factors. In the first category are global factors such as the

weight of the reconfiguration cost and the level of transition rates. They affect the number of distinct overlay topologies, as previously discussed, and therefore the number of clusters. In the second category are local factors that affect whether two given communication patterns should be put into the same cluster or separated into different ones. We identify three important local factors in our experiments. The first factor is the balance/imbalance of the occupancy time of two neighboring communication patterns in the communication requirement transition model. We find that the more unbalanced the occupancy time is, the more likely the two communication pattern are put into the same cluster. The intuitive explanation is that the system is reluctant to pay transition cost for the less occupied communication pattern and is prone to "absorb" the less occupied communication pattern into the dominant communication pattern and treat them as a whole. The second factor is the coupling between two communication patterns in the communication requirement transition model.

We define the *coupling* between communication patterns $\mathcal{C}_A$ and $\mathcal{C}_B$ as the sum of two quantities: the probability that the system reaches $\mathcal{C}_B$ before returning back to$\mathcal{C}_A$ once it leaves $\mathcal{C}_A$, and the probability that the system reaches $\mathcal{C}_A$ before returning back to $\mathcal{C}_B$ once it leaves $\mathcal{C}_B$. We find that the more coupled the two communication patterns are, the more likely they are put into the same cluster. By putting more coupled communication patterns in the same clusters, the system reduces the number of cross-cluster transitions therefore the reconfiguration cost. The third factor is the similarity between two communication patterns. The similarity is defined as the difference between the occupancy cost when the two communication patterns use their own minimum-occupancy-cost overlay topologies and that when the two communication pattern share the same overlay topology. We find that the more similar the two communication patterns are, the more likely they are put into the same cluster. By putting more similar communication patterns in the same cluster, the system can find an overlay topology that is more suitable for every communication pattern in the cluster which reduces the occupancy cost.

4.3.3 Summary

In summary, from the above case studies we understand the *structure of the optimal policy*: The optimal policies are composed of one or more optimal reconfiguration chains. The system steers itself into one of the optimal reconfiguration chains after a few steps of initial reconfiguration if it follows the optimal reconfiguration policy. The choice of optimal reconfiguration chain depends on the initial state of the system and makes a difference only when the initial reconfiguration cost is high. Within an optimal reconfiguration chain, communication patterns form clusters, defined by a distinct overlay topology in the chain. Clusters are overlapping when some communication patterns are associated with more than one overlay topology or exclusive when all communication patterns are associated with only one overlay topology. Most communication patterns, though, are associated with only one overlay topology. Topology reconfiguration is triggered only when the system makes a transition across clusters. The formation of clusters is affected by two categories of factors: global factors, such as the weight of reconfiguration cost and the level of transition rates, affecting the number of clusters, and local factors, such as level of balance of occupancy time, the coupling and similarity between two given communication patterns, affecting whether the two communication patterns are put into the same cluster.

From above case studies, we also learn the *structure of the problem* of constructing optimal reconfiguration policy, which roughly divides the whole range of β into three areas:

1. Lower Extreme Area: β is lower than a threshold β_0. The reconfiguration cost is trivial and the optimal policy is the one that always reconfigures the overlay topology to the one that minimizes the occupancy cost for the next communication pattern. We refer such a policy as the *Always Change Policy (ACP)*.

2. Upper Extreme Area: β is lower than another threshold β_1. The system is extremely conservative and always keeps its overlay topology unchanged (or at least after an initialization period). The reconfiguration cost is zero and optimal policy is one that minimize the overall occupancy cost in the session. We refer to this policy as the *Never Change Policy (NCP)*.

3. Middle Area: β is between β_0 and β_1. The system's behavior is fine tuned. This area can be divided into more threshold-based ranges. Simple policies, such as ACP and NCP policies, are not good and we need a more complicated method for constructing the optimal policy when the value of β is in this area.

4.4 Constructing Reconfiguration Policies for Large Systems

When the system has a large number of nodes or the process of transition among communication patterns is not Markovian, we cannot obtain the optimal reconfiguration policies by solving a Markov decision process. In this section, we propose methods for constructing good reconfiguration policies for these more general systems. We develop heuritic methods that reflect the properties and conform to the structures we observed in the optimal policies in Section 4.3.

4.4.1 Always-Change Policy and Never-Change Policy

The Always-Change Policy is simple: the system always reconfigures to the static-optimal topology for the next communication pattern whenever the communication pattern changes. The subroutine for finding the static-optimal topology is discussed in Section 3.2.

The Never-Change Policy never changes the overlay topology. Since the reconfiguration cost is zero, the optimal policy is one that minimizes the overall occupancy cost. Assume there are N distinct communication patterns in $X(t)$ and the percentage of occupancy time for communication pattern $\mathcal{C}_i$ is π_i, then the long-run average of the overall cost is:

$$\begin{aligned}\liminf_{\Delta t\to\infty}\frac{COST(\Delta t)}{\Delta t} &= \sum_{i=1}^{N}\pi_i\cdot f(\mathcal{C}_i,\overline{\mathcal{T}})\\ &= \sum_{i=1}^{N}\pi_i\sum_{u,v\in V}\mathcal{L}^{u,v}(\overline{\mathcal{T}})\cdot\mathcal{C}_i(u,v) = f(\sum_{i=1}^{N}\pi_i\mathcal{C}_i,\overline{\mathcal{T}})\end{aligned} \tag{13}$$

where $\overline{\mathcal{T}}$ is the common overlay topology in the NCP policy. Eq.13 shows that to minimize the long-run average of the overall cost, $\overline{\mathcal{T}}$ is actually the static-optimal topology for a virtual communication pattern $\sum_{i=1}^{N}\pi_i\mathcal{C}_i$ and can be found using the subroutine in Section 3.2.

4.4.2 Cluster-Based Policies

Inspired by the properties we observed in Section 4.3, we propose Cluster-Based Policies(CBP) as heuristic approximation to the optimal reconfiguration policy. The basic idea of constructing CBP policies is to mimic the structure of the optimal ones:

1. We group the communication patterns into non-overlapping clusters and find a common overlay topology for each cluster; the common overlay topology for a cluster is one that can minimize the overall occupancy cost for the whole cluster.

2. Whenever the transition in communication pattern happens, if the next communication pattern is in the same cluster, the system keeps the current overlay topology, otherwise it switches to the common overlay topology of the newly entered cluster.

The CBP policies maintain the essential structure of the real optimal policy but simplify it in the following terms: first, there is only one optimal reconfiguration chain in the CBP policies; second, every communication pattern is associated with only one overlay topology (and therefore only one cluster) and all clusters in CBP policies are non-overlapping. The simplification allows us to construct the CBP policies efficiently. Performance evaluation in Section 4.5, however, shows that CBPs are almost as good as the real optimal policies.

Fig. 21 sketches the algorithm of constructing CBP policies. The algorithm needs three types of input about $X(t)$: the set of distinct communication patterns, $\{\mathcal{C}_i | 1 \leq i \leq N\}$; the average percentage of occupancy time that the system runs with $\mathcal{C}_i$, denoted by π_i; and the average number of transit from $\mathcal{C}_i$ to $\mathcal{C}_j$ per unit of time, denoted by b_{ij}. The value of π_i's and b_{ij}'s can be estimated in a real system via statistical modeling methods. Specially, for applications in which the transitions between communication patterns can be modelled with Markov or Semi-Markov processes, π_i's and b_{ij}'s can be derived from the transition model by calculating the stationary probabilities of the processes.

As most k-means style clustering algorithms, the proper number of clusters needs to be estimated beforehand, based on the weight of reconfiguration cost. The proper number can also be find by trying out all numbers in a certain range (e.g., in worst case, from 1 to

the total number of communication patterns) and choosing the one that generates the best result.

Assume we are clustering N communication patterns into L clusters. The algorithm starts by randomly assigning the N communication patterns into L clusters. During each iteration, the algorithm reassigns each communication pattern to the cluster that is most "suitable" for it. The iterations stop when the clustering converges and the policy corresponding to the final clustering is returned as the result of the algorithm.

PROCEDURE CONSTRUCT-CBP-POLICY
Initialize-Clusters :
  randomly assign $\mathcal{C}_1$, $\mathcal{C}_N$, ... $\mathcal{C}_N$ to S_1, S_2, ... S_L ;
Adjust-Clusters :
  for each communication pattern $\mathcal{C}_i$, $1 \leq i \leq N$:
    find the most suitable cluster S^* for $\mathcal{C}_i$:
      for each cluster S_j, $1 \leq j \leq L$:
        temporarily move $\mathcal{C}_i$ to S_j;
        convert the clustering (S_1, S_2, ... S_L) to policy P; — line (a1)
        evaluate the cost of policy P; — line (b)
      Let $\mathcal{S}^*$ is the clustering with the least cost in the above loop;
      move $\mathcal{C}_j$ to S^*;
Loop-Back :
  go back to Adjust-Clusters until the clustering does not change;
Post-Handle:
  convert the clustering (S_1, S_2, ... S_L) to policy P; — line (a2)
  return policy P;

Figure 21: Pseudo-code of Constructing Cluster-Based Polices

In Fig. 21, lines (a1) and (a2) involve a procedure that converts a clustering to its corresponding cluster-based reconfiguration policy and line (b) involves a procedure that calculates the cost of the policy. We describe the two procedures in detail in the remainder of this section.

Converting Clustering To Policy The major task in this procedure is to find one common overlay topology for each cluster. Once the common overlay topologies are decided, the policy is fully defined by the following rule: whenever the system makes a transition between communication patterns within the same cluster, the overlay topology is not changed; whenever it makes a transition across clusters, it reconfigures the overlay topology to the

common overlay topology for the newly entered cluster.

The common overlay topology for a cluster is one that can minimize the total occupancy cost of the cluster. Formally, consider a cluster containing communication patterns $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s$, the common topology of the cluster is a overlay topology $\mathcal{T}$ that minimizes the occupancy cost of the whole cluster: $COST_c(\Delta t) = \sum_{i=1}^{s} \pi_i \Delta t \cdot f(\mathcal{C}_i, \mathcal{T})$. Similar to the logic in Section 4.4.1, this optimal common overlay topology is actually the optimal-static topology for a virtual communication pattern $\sum_{i=1}^{s} \pi_i \mathcal{C}_i$ and can be found using the subroutine in Section 3.2.

Calculating Policy Cost Assume $\mathcal{T}_i$ is the topology assigned to communication pattern $\mathcal{C}_i$ by the policy. Because in a CBP policy, each communication pattern is associated with only one overlay topology, we have: $\liminf_{\Delta t \to \infty} \frac{COST_p(\Delta t)}{\Delta t} = \sum_{i=1}^{N} \pi_i f(\mathcal{C}_i, \mathcal{T}_i) + \beta \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} g(\mathcal{C}_i, \mathcal{T}_i, \mathcal{C}_j, \mathcal{T}_j) \cdot b_{ij}$.

4.5 Performance Of Approximate Policies

4.5.1 Performance of Constructed Policies — Small Networks

Table 1 shows the actual experimental parameters we use to generate a typical case problem.

First, we observe how the resulting clustering and the performance of corresponding CBP policies are affected by the number of clusters input to the clustering algorithm described in Fig. 21. The experiment results from the problems constructed using the parameters in Table 1 are shown in Fig.22, while figures (a) and (b) are for two different weights on reconfiguration cost (β in Eq. 11). The x-axis represents the number of clusters, K, that we input to the algorithm. The right y-axis represents the actual number of *non-empty* clusters in the resulted clustering; the figure shows that clustering algorithm does not necessarily use up all the clusters if it finds that less clusters give better result. The right y-axis represents the cost of resulting cluster-based policy. The figures show that a good estimation of the proper number of clusters can help the algorithm to find the best cluster-based policy whose cost is most close to that of the optimal policy.

Fig. 23.a compares the cost of the optimal policy, the ACP policy, the NCP policy and the best CBP policy, whose cost is most close to that of the optimal policy, for the

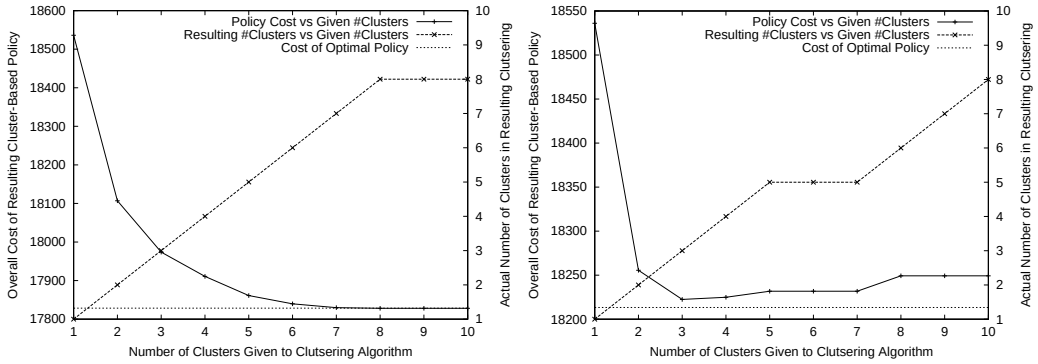
parameters	small network	large network
underlying native network	-	Internet graph generated using GT-ITM, 1400 nodes: 1 transit domain(200 nodes), 20 stub domains(60 nodes in each stub)
number of overlay nodes	5	40, randomly selected from the stub domains
overlay link cost matrix	random value in range [10,15]	number of hops in native network
number of communication patterns	10	10
number of transitions from each communication pattern	random integer in range [1,3]	random integer in range [1,3]
transition rates between communication patterns	random value in range [2,6]	1
data demand between pairs of overlay nodes in communication patterns	random value in range [0.1, 100]	1 for 15 percent of total pairs (randomly chosen), 0 for other pairs
degree bound on feasible overlay topologies	2	4
number of feasible overlay topologies	72	-

Table 1: Experiment Parameters

same case constructed using the parameters in Table 1. The y-axis represents the policy cost. The x-axis represents the spectrum of the weight assigned to the reconfiguration cost. The figure shows that the cost of the NCP policy does not change when the weight of reconfiguration cost is increased because the policy operate with a single policy and does not incur a reconfiguration cost. The cost of the NCP policy coincides with that of the optimal policy for a large β, which means that NCP policy is actually optimal when the weight of the reconfiguration cost is beyond a certain threshold. The figure also shows that the ACP policy is actually optimal when the weight of the reconfiguration cost is below a certain threshold but its cost increases very quickly when the weight of the reconfiguration cost is large. Finally, the figure shows that the CBP policies approximate well the optimal policy when the weight of the reconfiguration cost is in its middle range.

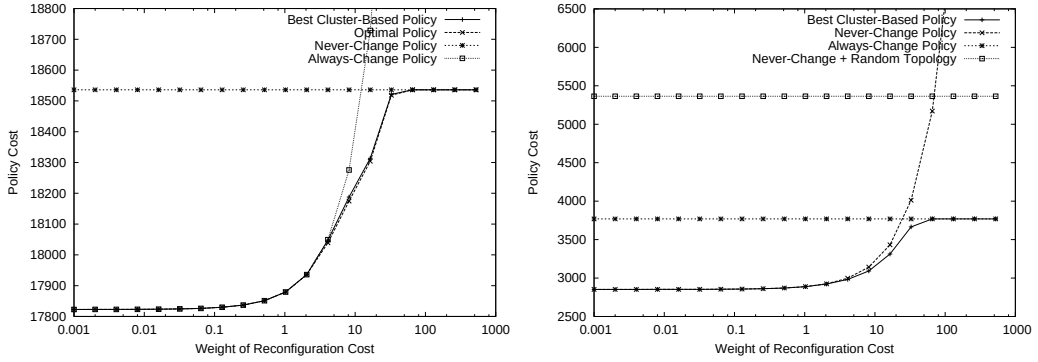
4.5.2 Performance of Constructed Policies — Large Networks

We also experiment on problems with larger number of nodes. We generate a native network using GT-ITM topology generator [15] and select some native nodes from the stub domains as overlay nodes. We assume the transitions between communication patterns are



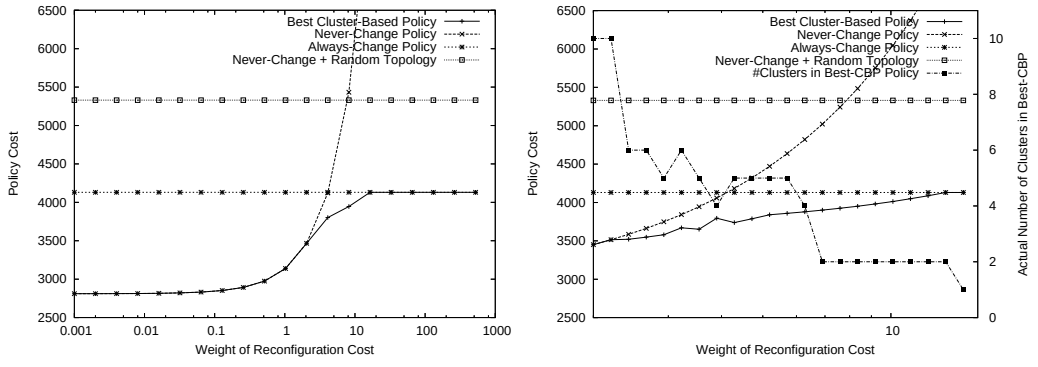
(a) $\beta = 0.1$ (b) $\beta = 10$

Figure 22: Effect of Given Number of Clusters



(a) Experiment With 5 Nodes (b) Experiment With 40 Nodes

Figure 23: Performance of Approximate Policies Against Optimal Policies



(a) (b)

Figure 24: Performance of Approximate Policies Against Optimal Policies in Non-Markovian Cases

Markovian. Table 1 shows the actual experimental parameters we use to generate a typical case problem.

Fig. 23.b presents the costs of different policies for this problem. The plots of NCP, ACP and best-CBP are very similar to those in Fig. 23.a. The figure also presents the cost of a naive policy that random chooses an feasible overlay topology and sticks with it, a policy that is actually used by many overlay applications. The cost is the mean over 20 random feasible overlay topologies. The figure shows that NCP, ACP and CBP policies perform much better than the naive policy in their applicable ranges.

Fig. 24.a presents the performance of NCP, ACP, best-CBP and the naive policy for a non-Markovian problem. Most parameters for the experiment are the same as those given in Table 1. The transitions between communication patterns, however, are semi-Markovian this time: we first randomly assign the transition probabilities in the embedded Markov chain, and then, for each transition, we assign the average pre-transition occupancy time to a random value in the a certain range, e.g., $[0.2, 1]$.

The plots in the figure are very similar to those in Fig. 23.b: the NCP, ACP and CBP policies perform much better than the naive policy in their applicable ranges and the CBPs adapt well when the weight of reconfiguration cost varies. Fig. 24.b magnifies Fig. 24.a for the portion where β is in range [2,16] and CBPs outperform both NCP and ACP. Both the policy-cost and the number-of-cluster plots for the best-CBPs show general trends similar to those in Fig. 19. This reaffirms our belief that the way we construct CBPs reflects the basic structures of the optimal policies and grasps the essential tradeoff between the occupancy cost and the reconfiguration cost inherent in the dynamic overlay topology reconfiguration problems. Note that there are "ripples", however, in both plots. This is due to the fact that the constructed CBPs are merely approximate alternatives to the optimal policies and the fact that our clustering algorithm and simulated annealing algorithm are also approximate. The ripples, however, are minor; the constructed CBPs perform consistently in approximating the optimal policies.

In our policy construction methods, the formulation of function $f(\cdot)$ in Eq.10 can be substituted with other formulations without affecting the applicability of the algorithms. Fig. 25

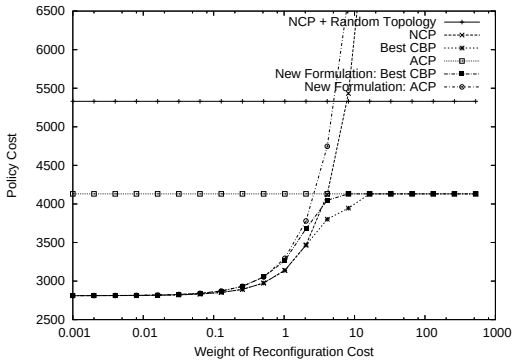


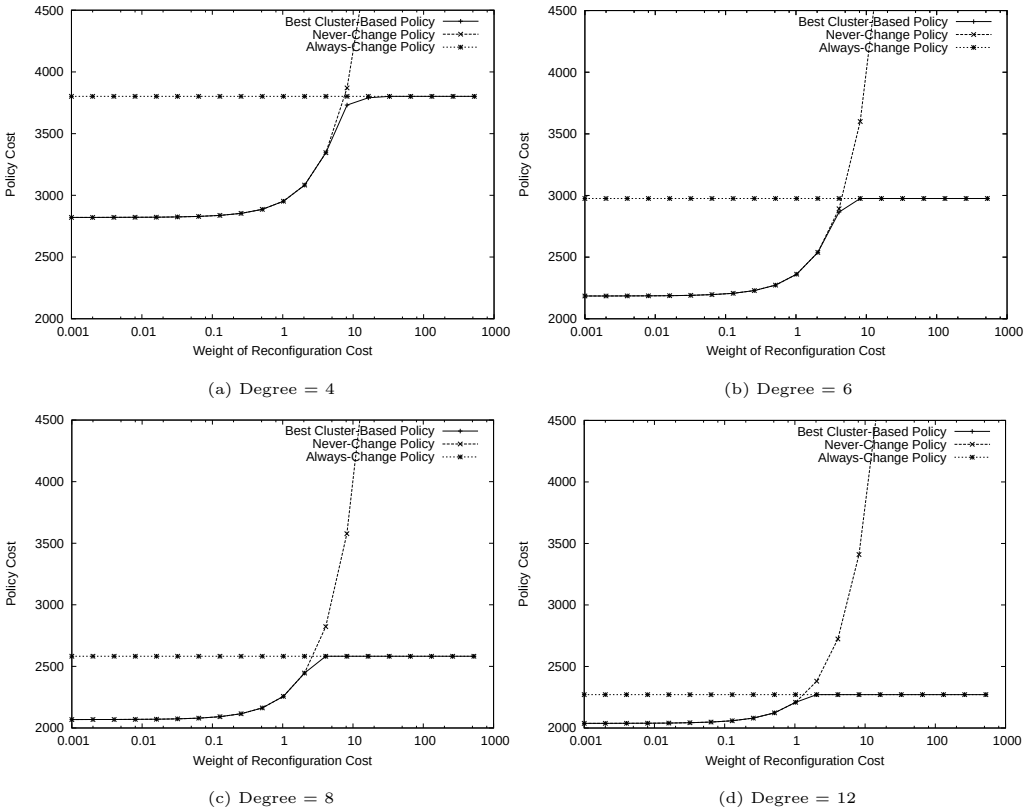
Figure 25: Using Other Formulations of Reconfiguration Cost

shows the cost of resulting policies when another formulation $g(\mathcal{C}_{old}, \mathcal{T}_{old}, \mathcal{C}_{new}, \mathcal{T}_{new}) = \sum_{u,v} w(u,v) \cdot |\mathcal{T}_{old}(u,v) - \mathcal{T}_{new}(u,v)|$ is used, where $w(u,v)$ weights the overlay link (u,v) with the number of end-to-end minimum-operation-cost paths that cross the link in topology $\mathcal{T}_{old}$. For NCP and random-NCP policies, the use of a new formulation does not affect the resulting policies and their cost. For ACP, the policies themselves are not affected; their costs changed, reflecting the new formulation, but the difference is trivial when β is in its lower range (where ACP policies are applicable). So only CBPs are affected. Fig. 25 compares the experiment results from the new formulation (Formulation 2) to those in Fig. 24.a (the two sets of experiments use same parameters except for the formulation). The figure shows that the CBPs still tune smoothly under the new formulation. They generally become less aggressive though, because the new formulation numerically generates higher reconfiguration cost than the old one.

4.5.3 Implications on Overlay Network Design

In this section, we present some additional results we observe when we apply our policy construction methods to large systems. Our purpose of presenting these results are two-fold. First, the results might be interesting by themselves to some overlay system designers. Second, the results show the usefulness of our policy construction methods in studying overlay systems in general — the results are not easily observed without studies on large systems.

Fig. 26 shows how the degree of an overlay network affects the cost of the reconfiguration



(a) Degree = 4

(b) Degree = 6

(c) Degree = 8

(d) Degree = 12

Figure 26: Degree of Overlay Networks versus Cost of Reconfiguration Policies

policies. Most parameters for the experiment are the same as those given in Table 1. The transitions between communication patterns are Markovian. From each communication pattern, the system can make up to 4 transitions and the transition rates are random values in range [1,5].

The four figures show the cost of the NCP, ACP and best-BCP policies when we set the degree bound of the overlay network to 4,6,8, and 12, respectively. We have three interesting observations from these experiments. First, the cost of the policies decreases when the degree bound increases. Intuitively, on one hand, with larger degree bound, there are more feasible overlay topologies; the system may be able to find better overlay topologies with lower occupancy cost. On the other hand, with larger degree bound, the system may be able to establish new overlay links without having to tear down the old ones; this reduces the reconfiguration cost. This suggests the overlay network designers use larger degree bound when possible. Second, the gain of using a larger degree bound varies. In Fig. 26, when the degree bound is increased from 4 to 6, both the cost of the NCP and that of the ACP decrease significantly. When the degree bound is increased from 6 to 8, only the cost of the NCP decreases significantly. After that, neither cost decreases significantly when the degree bound is increased. This suggest that overlay network designers should not pursue larger degree bounds blindly. The benefit may be limited, especially when the weight of reconfiguration cost is at its low range (where the ACP is optimal). Finally, pay attention to the space below the plot of the best-CBP and above the plot of the ACP in the figures. The space indicates how much the CBPs can outperform the ACP and the NCP in their applicable range. Notice that the space becomes narrower when the degree increases. This suggests that the CBPs outperform ACP and NCP most when the overlay networks have a very restrictive degree bound.

The above results belong to the domain of a wider class of overlay network design problems, e.g., how do we select overlay nodes from a set of native nodes? how many nodes does an overlay need? what is the appropriate degree bound for an overlay? Our results presented here are far from complete and thorough. The topics stretch beyond the scope of this thesis and deserve much more research in the future.

4.6 Summary

We have studied the problem of dynamically reconfiguring the topology of an overlay network in response to the changes in the communication requirements. We have identified the relevant factors in a service overlay network that affect the cost of using the overlay. We have considered two costs of using an overlay: the occupancy cost and the reconfiguration cost. The ideal goal is to find the optimal reconfiguration policies that can minimize the potential overall cost of using an overlay. The problem is NP-hard and good approximate policies are called for. We have studied the structure of the problem, as well as the properties and structures of the optimal policies, through experiments on small systems with Markovian properties, where the problem can be modeled as a Markov decision process and solved efficiently using Howard's policy-iteration algorithm. We then used our observations as heuristics and proposed methods of constructing approximative policies that reflect the properties and structures of the optimal policies. We have shown that our methods are applicable to large general systems and the constructed policies achieve good performance. We have also shown that our methods are helpful for studying other overlay design problems.

CHAPTER V

COST-EFFECTIVE CONTENT RESILIENCY SERVICE

5.1 Introduction

An ounce of prevention, a pound of cure — resiliency strategies can help to improve information and service availability in the face of failures by reducing either the probability of failures or their impact. Such resiliency strategies include, for example, introducing fault-tolerant hardware, using backup paths in network routing, and replicating data or service offering at multiple locations. On one hand, these resiliency strategies increase the expected information or service availability and, thus, reduce the potential damage caused by data loss or service interruption. One the other hand their deployment incurs a *prevention cost*, e.g., additional hardware, monitoring infrastructure, storage, computation and communication resources. In the most general terms, a cost-effective resiliency strategy is one that considers both the benefit and cost of deploying a protection mechanism and choose the optimal tradeoff between the benefit and cost.

While engineers have invested a significant amount of effort in improving the reliability of computer hardware and software, failures are still common to today's computer systems and networks. Failures can be caused by not only worn-out devices and software bugs but also external factors such as power outage, environmental conditions and operational accidents. When the computer and network system that stores a piece of data fail, the data becomes unavailable to its users until the system is recovered. Replicating the same content to multiple content servers (e.g., data centers) at different locations is one of the most important resiliency strategies for data protection in the presence of failures[35][90][25]. It can significantly improve the availability of the data even when multiple servers fail in the same time, and thus reduce the potential damage caused by the loss of data.

While content servers for resiliency purpose can be deployed by an organization for its private use, in this thesis we are most interested in *Content Resiliency Service Networks*,

a pool of content servers purposely deployed by a third-party *Content Resiliency Service Provider* that provides resiliency service for service users by replicating data uploaded by the users to some other content servers. As shown in Fig. 4, a content resiliency service provider deploys a pool of tens or hundreds of content servers at different locations over the Internet. Users at the end-systems access (i.e., upload to and download from) their data at a local content server. The local data servers are then responsible to replicate the data to some of other content servers in the pool so that the data is available when the local content servers or even part of the replicating content servers fail at the same time. On one hand, a content resiliency service provider contracts with service users in the form of service agreement, which allows the users to specify the desired level of content resiliency or the penalty of loss of data. On the other hand, the content resiliency service provider pays *prevention costs*, such as the communication cost of delivering data from its origination local content server to the remote replicating content servers, the storage cost of storing the data, and various types of management costs, to maintain the normal operation of the content servers and achieve the level of content resiliency previously agreed between the service provider and the users.

Similar to the case of service overlay network that is discussed in Chapter III and IV, there are configuration and dynamic reconfiguration issues in content resiliency service networks, particularly, the choice of replication locations for a piece of data from the pool of available content servers. Fig. 5 shows a content resiliency service network with a pool of nine content servers, represented by the circles. A user, represented by the square box, uploads its data to its local content server S_0, and S_0 is responsible to choose a set of remote replication locations from the other eight content servers and deliver the data to the chosen set of servers. For example, it can choose servers S_3 and S_7 and form a replication set $\{S_0, S_3, S_7\}$, or choose servers S_2 and S_5 and form a replication set $\{S_0, S_2, S_5\}$.

Intuitively, the farther the servers replicating the same data reside from the content origination and from each other, the more unlikely the servers will fail in the same period but the more communication cost will be incurred. A good configuration of content server

replication relationship should maintain a target level of content resiliency while minimizing the prevention cost incurred by the configuration and avoid excessive costs caused by over-reaction to potential risks. Furthermore, when some content servers actually fail and recover, the replication sets need to be dynamically adjusted to maintained the desired level of content resiliency.

One critical component in the development of such a cost effective configuration is the assessment of failure risks for a given environment. The probability that all the content servers in the replication set fail at the same time must be quantified. Without such quantification, the risks associated with a particular set of replicating content servers cannot not be evaluated and any effort trying to reduce the prevention cost is clueless. When failures are local to each content servers and therefore independent across the servers, such quantification is relative easy to achieve. However, in contrast to many researchers' assumption that failures are independent to each other, many failures (e.g., those caused by power-outages, weather and environmental disasters, and worms) are correlated to each others. Components in modern distributed systems are becoming more interconnected and interdependent, and, consequently, may more frequently be simultaneously impacted by the same underlying fault event. This makes the assessment of failure risks a more complex task. While modeling independent failures is relatively easy, modeling of correlated failures is much more challenging. The obvious way of describing correlated failures is to specify the probability of simultaneous failure for each subset of nodes. Some models enumerate all possible system states and calculate transition probabilities between them. These models, however, are difficult to apply in terms of representation and instantiation due to the exponential number of states. More succinct yet meaningful models are called for. Researchers have developed various approach for modeling correlated failures [90, 5, 65], but these models do not allow for deriving the probability of simultaneous failures for different subsets of entities in a practical, consistent and intuitive way. A new approach for modeling correlated failure needs to be develop before the configuration and dynamic reconfiguration issues in content resiliency service networks can be tackled.

In addition to risk assessment, developing cost effective resiliency strategies also requires

understanding the structure of the prevention cost. Various non-trivial components contributing to the prevention cost should be taken into account. Furthermore, for systems that have risk or resource constraints that evolve in time and need dynamic reconfiguration of resiliency strategies, the cost-effectiveness of the strategies should be measured based on their long-term achievement instead of short term benefits.

In this thesis we study the problem of cost-effective data replication in distributed systems impacted by both *independent* and *geographically correlated* failures. Geographically correlated failures are those caused by events that impact nodes within a geographical vicinities, e.g., power-outages, weather and environmental disasters, catastrophic events(terrorist attacks, military actions). Some types of *network correlated* failures and *administration correlated* failures can also be approximated as geographically correlated under certain conditions. The impact of geographically correlated failures on system resiliency has been studied in the literature, in both industry and academia (e.g., [35][90][25]); solutions relying on remote placement of system components have been proposed. While such efforts are helpful in improving overall system resiliency, they suffer from the lack of good models for geographically correlated failures. Without such models, the improvement in system resiliency cannot be fully quantified and measures of cost efficiency cannot be developed.

Our research work presented in this chapter is twofold. First, we present a new approach for modeling correlated failures. Instead of modeling entity correlations directly, our methodology projects entities to a Euclidean space based on a *distance* metric between entities; entity failure correlations are derived through the modeling of all possible fault events within this space. This approach is then applied to model geographically correlated failures. This model allows for the computation of data availability in replication systems against both independent failures and geographically correlated failures. Second, using the failure model, we study the problem of selecting the replication locations from the pool of content servers. Specifically, we consider where replicas should be placed so that the prevention cost is minimized while at the same time a desired level of data availability, in the presence of both independent failures and geographically correlated failures, is guaranteed. Furthermore, we consider the dynamic conditions of the content servers over time

and propose dynamic replication policies that reduce the long-term operation cost of the system by taking into account both the reconfiguration and static prevention cost.

The remainder of this chapter is organized as follows. In Section 5.2, we describe a new approach for modeling correlated failures. The approach is demonstrated using a model for geographically correlated failures in Section 5.3, where we also formulate the static configuration problem in content resiliency service network after identifying cost factors in the replication process. In Section 5.4, we present optimization algorithms, including ones that generate the real optimal solution and heuristic ones, for solving the static configuration problem in content resiliency service networks. We also evaluate the performance of the heuristic algorithms at the end of the section. In Section 5.5, we discuss the dynamic configuration problem in content resiliency service networks and present several dynamic reconfiguration policies that can achieve cost-effectiveness when system conditions change over time.

5.2 A General Approach for Modeling Correlated Failures

Potential sources of failures can be generally classified in two main categories; *independent* and *correlated*. In distributed systems that are composed of multiple entities, independent failures are those that cause a single entity failure, without any impact on other system entities. They include, for example, entity-specific hardware and/or software failures. In contrast, correlated failures are caused by fault events impacting a set of one or more entities simultaneously or within a very short time interval. Multiple types of failures fall within this description. These include failures caused by *geographically correlated* fault events, such as those caused by power-outages, weather and environmental disasters, catastrophic events (terrorist attacks, military actions); caused by *network correlated* fault events, such as router failures, DoS attacks causing local congestion, and worms; caused by *social network correlated* fault events, such as virus blasts using email as the propagation medium, and *administration correlated*, such as the malfunction of devices in an administrative domain caused by the same (human) operation mistake. In this thesis, it is not our intention to model every types of correlated failures. Instead, we seek a common approach that can be

used generally to model the cause and impact of various types of failures.

The fundamental objective of modeling correlated failures is to provide the capability of calculating the probability of simultaneous failures for different subsets of entities. Due to the very large number of subsets and the multiple dependencies among entities, any model that specifies the probability of simultaneous failure for every possible subset of entities is likely to be complex and of limited usability. Research work has focused on building models that express only the two-way correlations between entities (e.g., conditional probabilities that one entity fails given another entity fails) and approximate multiple-way correlations using two-way correlations. Weatherspoon et al. [90] cluster entities based their two-way correlations and use the clusters as an implicit measure of multiple-way correlations in system design. This method, however, does not produce quantifiable models and, hence, cannot be used to develop systems with a prescribed availability requirement. Approaches to building models that can approximate multiple-way correlations from two-way correlations in a quantifiable way are proposed in [5][65]. These models, designed to capture failures of m-out-of-n entities, do not allow for calculating the probability of failure for specific subsets of entities. More importantly, these approaches provide limited intuition about the nature and the cause of correlated failures and this impairs the accuracy and consistency of entity correlation in the resulting models.

In this thesis, we propose a new general approach to modeling correlated failures: instead of modeling failure correlations between entities *directly*, we model the occurrence of fault events that impact the entities and cause correlated failures. Failure correlations between entities are consequently *implicitly* inferred. The approach has two steps: first, entities are projected to a Euclidean *fault field* and assigned coordinates based on a certain distance metric. Depending on the type of correlated failures under consideration, this distance metric could be a physical one, such as geographic distance, a logical one, such as network distance and distance in social network or administration hierarchy, or simply the measured two-way failure correlation between two entities. If the distance does not satisfy the triangular-inequality, the non-Euclidean space can be converted to a Euclidean space, possibly with additional dimensions. Second, fault events are modeled over the fault field.

Specifically, the location of faults and their impact are modeled. Thus the fault field acts as a medium to establish correlation between the fault events and the entities using their locations in the fault field. Models constructed in this way intuitively reflect the nature and cause of correlated failures and provide a consistent view of failure correlation between entities.

In Section 5.3.1 we describe the an example use of our approach by modeling *geographically correlated failures* in content resiliency service networks. The model serves for two purposes. First, we use it to demonstrate the feasibility and applicability of the approach described above. Second, the model is used throughout the remaining sections of this chapter to study cost-effective configuration issues in content resiliency service network in the presence of both independent and geographically correlated failures. Note that for geographically correlated failures, the fault field naturally lies on a 2-dimensional plane and a server's geographic location serves as its coordinates in the fault field. For other types of correlated failures, the construction of the fault field could be more complex (e.g., involving the conversion of a non-Euclidean space to a Euclidean one) and the model for fault events could be different. In Section 6.2.2 we will give more discussion how different models can affect the results presented in the thesis.

5.3 Static Configuration Problem in Content Resiliency Service

In this section, we formally discuss the static configuration problem in content resiliency service network, the problem of choosing cost-effective replication sets in a pool of content servers in the presence of both independent failures and geographically correlated failures. We first describe our failure models for independent failures and geographically correlated failures and discuss the meaning of data availability under such failure models. We then identify the cost factors in the data replication process and formulate the problem of cost-effective data replication as a constrained minimization problem. We use discrete time models in which the time axis is divided into slots. In real systems, the length of the time slot depends on the system's response time to failures. We assume that a system has enough

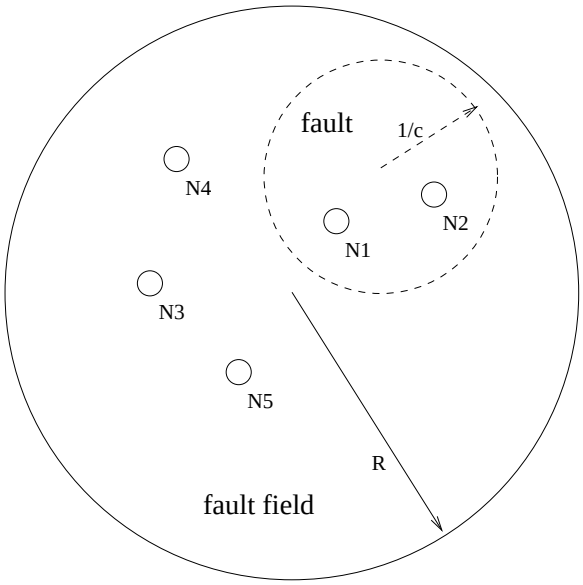


Figure 27: Geographically Correlated Failure Model

time to detect and complete its responses(e.g., recovering data, adjusting backup sets, etc) by the end of a time slot for all failures that have happened during that time slot. In this section and Section 5.4, we assume failed servers will be fully repaired and come back to service by the beginning of next time slot and therefore focus our discussion on a single time slot. The assumption will be relaxed in Section 5.5 when we discuss cost minimization over a sequence of time slots under dynamic system conditions.

5.3.1 Failure Models

We assume the pool of content servers in the content resiliency service network are distributed over a 2-dimension fault field. We assume the geographic location of each content server is known to all other content servers (e.g., by an administrator or through some information exchanging protocol) in the form of a coordinate in the 2-dimension fault field. We consider two categories of failures that could happen to the content servers: independent failures and geographically correlated failures.

Geographically Correlated Failure Model Instead of directly specifying the probability of content server failures and their correlation, we model the geographical distribution and the impact of geographical fault events (e.g., power outage) that could cause multiple

content servers fail simultaneously in the same time slot.

First, we assume that in each time slot, the probability that one geographical fault event occurs is P_{corr} and the probability that more than one such fault events occur is ignorable. Second, we assume that the center of a fault event is distributed according to a uniform random distribution over a circular *fault field* with a radius R on a 2-dimension plane. For simplicity of presentation, we use the center of the circular fault field as the origin of the 2-dimension space of the fault field and adjust all coordinates accordingly. Third, to characterize the phenomena that fault events affecting large geographical areas are much rarer than those affecting small geographical areas, we assume the impact of a fault event is exponential decaying and model it as follows: the impact of a fault event reaches as far as a distance r from the center of the fault, where r is a random variable that is exponentially distributed with a parameter c; when a fault event happens, all content servers within a radius of r to the center of the fault event fail simultaneously.[1]

Fig. 27 shows a visual presentation of our model for geographically correlated failures. The fault field is contained within the solid-line circle and the impact of an example fault event is represented with a dashed-line circle. Among the five content servers ($N1$-$N5$) in the figure, two content servers (N_1 and N_2) fail simultaneously because of the example fault event.

Independent Failure Model In addition to correlated failures, we also consider independent failures. They are caused by factors local to each specific content server, e.g., broken hardware and crashed software. In this section, we assume that each content server fails independently with a fixed probability P_{ind} in each time slot.

The objective of developing a cost-effective configuration of replication locations is to reduce the occurrences of data loss as well as the cost of replicating data. Data loss happens when both the content server that originally receives a piece of user data and all the content servers that replicate that piece of data fail in the same time slot.

[1]Deepak Ganesan et al. also suggested the use of circles to characterize the impact of geographically correlated failures [35], but they used fixed-size circles without capturing the phenomena that large-impact fault events are much rarer than small-impact ones.

5.3.2 Analysis of Data Availability

When a fault event occurs at center v, all content server in a replication set S will fail correlatively due to this fault even if and only if their maximum distance to v is equal or less than the radius of the fault event. This at, the probability that they all fail in this fault is

$$\mathcal{F}_{corr}(S,v) = \int_{r=d(S,v)}^{+\infty} \frac{e^{-cr}}{c}\, dr = e^{-c\cdot d(S,v)} \tag{14}$$

where $d(S,v) = \max_{N_i \in S} d(N_i, v)$. So if the probability of independent failures is zero, the probability that all content servers in S fail in the same time slot is the integral of Eq. 14 over all possible centers of this fault event (according to fault location distribution model given in Section 5.3.1 in the fault field. That is,

$$\mathcal{F}_{corr}(S) = \frac{P_{corr}}{\pi R^2} \int_v \mathcal{F}_{corr}(S,v)\, dv \tag{15}$$

When both independent failures and geographically correlated failures are possible, the simultaneous failure of a whole set of content servers during the same time slot can be the result of various combinations of independent failures and correlated failures (i.e., part of the set fail independently and part of the set fail correlatively.) First, consider the conditional probability that all content servers in S fail in a time slot given that a fault event has happened in the time slot with its center at v. To calculate this conditional probability, we need to consider separately the cases where the radius of the fault event covers one server, or two servers, ..., or all servers in the set S. Assume the content servers in S have been sorted based on their distances to the center of fault, v, and they are $N_{i_1}, N_{i_2}, \ldots, N_{i_{|S|}}$ in increasing order, we have

$$\mathcal{F}_{comb}(S) = Prob\{r \geq d(N_{i_{|S|}}, v)\} + \sum_{s=1}^{|S|} Prob\{r \in [d(N_{i_{s-1}}, v), d(N_{i_s}, v))\} \prod_{k=s}^{|S|} Prob\{N_{i_k} \text{fails}\}$$

where we set $d(N_{i_0}, v) = 0$. That is,

$$\mathcal{F}_{comb}(S,v) = e^{-c\cdot d(N_{i_{|S|}},v)} + \sum_{s=1}^{|S|} (e^{-c\cdot d(N_{i_{s-1}},v)} - e^{-c\cdot d(N_{i_s},v)}) {P_{ind}}^{|S|-s+1} \tag{16}$$

Therefore the probability that all content servers in S fail in the same time slot is,

$$\mathcal{F}_{comb}(S) = (1 - P_{corr}){P_{ind}}^{|S|} + \frac{P_{corr}}{\pi R^2} \int_v \mathcal{F}_{comb}(S,v)\, dv \tag{17}$$

Note that there is no close form for the integral but it can be approximated using numeric methods.

When a user of the content resiliency service uploads its data to a local content server N_A, N_A choose a subset S_A from the pool of content servers as its backup set and replicate the data to each content server in the backup set. Thus N_A and S_A together form the replication set for the uploaded data. The *availability* of the data is thus defined as the probability that either N_A or at least one of the content servers in its backup set S_A survives the risk of both independent failures and geographically correlated failures through a time slot. That is, the availability, denoted by $\mathcal{A}(N_A \cup S_A)$, equals $1 - \mathcal{F}_{comb}(N_A \cup S_A)$.

5.3.3 Cost-Effective Configuration of Replication Locations

Once the backup set for a content server is configured, the replication process involves two types of cost: the communication cost and the storage cost. The former is the cost incurred when moving data from its origination local content server N_A to all the content servers in N_A's backup set S_A. The latter is the cost of storing data at its local content server and the backup content servers. Considering the availability of cheap large volume storage devices, we assume in this thesis the storage cost is trivial and focus on the communication cost.

We model the communication cost of moving one unit of data from a content server N_A to another content N_B with the following equation:

$$\mathcal{C}(N_A, N_B) = b_1 \cdot d(N_A, N_B) + b_2 \cdot d^2(N_A, N_B) \tag{18}$$

where $d(N_A, N_B)$ is the Euclidean distance between N_A and N_B. The cost function is composed of a linear term and a quadratic term. The constant factors b_1 and b_2 are coefficients for the two terms, respectively, and their values depend on the nature of the communication network. The using of two terms is an approximation for today's communication network that combines various types of communication media and methods. For example, in wired networks, the linear term is most significant, and in wireless networks, the quadratic term also plays a significant role.

Note that in the above cost model, we assume communication cost is purely a function of geographical distance between the content servers. This may not the most accurate model

for modeling communication cost in the real systems. A more general model is a cost matrix that characterizes the communication cost between every pair of content servers, as the one described in Section 3.2.1 where we discuss service overlay networks. We will discuss other variation of communication cost models and their impact later in Section 6.2.2.

For simplicity, we assume during each time slot, a content server receives one unit of data from the local users. Then the total communication cost of moving data received by a content server N_A in one time slot from N_A to its backup set S_A is

$$\mathcal{C}(N_A, S_A) = \sum_{N_B \in S_A} \mathcal{C}(N_A, N_B) \tag{19}$$

Again, we use a relative simple communication cost model here by assuming data are delivered between content servers using unicast. In Section 6.2.2, we will discuss other communication models that allow content servers to relay traffic during replication process.

Given $S_{candidate}$, the pool of all content servers in a content resiliency service network that can serve as a backup server for N_A, the objective of choosing an optimal static configuration is to determine an optimal subset of $S_{candidate}$ as the backup set for N_A: the communication cost is minimized under the constraint that the availability of the data received by N_A is maintained over a target lower bound A_l that is specified in a user's service agreement. That is,

$$\begin{aligned} \min \quad & \mathcal{C}(N_A, S_A) \\ \text{s. t.} \quad & S_A \subseteq S_{candidate} \\ & \mathcal{A}(N_A \cup S_A) \geq A_l \end{aligned}$$

The problem is combinatorial and the explosion of solution space makes naive exhaustive search impractical. In Section 5.4, we first propose a branch-and-bound algorithm which can practically find the optimal backup sets for problems with up to around 30 candidate content servers. With its help, we then do case studies to observe the properties of the optimal backup sets. Based on the observation collected in the case studies, we develop heuristic-based algorithms which allow us to handle problems with even larger scale. We will describe the operation of the algorithms from the point of view of a local content

Algorithm 1 Branch-and-Bound

1: $S_0 = \{N_0\}$
2: $S_{full} = S_0 \cup \{N_1, N_2 \ldots, N_k\}$
3: set the root of tree $\mathcal{T}$ to S_0
4: let cost upper bound $C_u = +\infty$
5: let $S_{optimal} = null$
6: **while** $\mathcal{T} \neq \phi$ **do**
7: randomly choose a tree-node S_{curr} from tree $\mathcal{T}$
8: **if** $S_{curr}.cost \geq C_u$ **then**
9: delete from tree $\mathcal{T}$ the subtree rooted at S_{curr}
10: **else if** $S_{curr}.availability \geq A_l$ **then**
11: let $C_u = S_{curr}.cost$
12: let $S_{optimal} = S_{curr}$
13: delete from tree $\mathcal{T}$ all nodes whose costs are equal or greater then C_u
14: **else if** S_{curr} is expandable **then**
15: Branch out a new child S_{new} below S_{curr} in tree $\mathcal{T}$, where S_{new} belongs to $\{S_{curr} \cup \{N_i\} \mid N_i \in S_{full} - S_{curr}\}$. Once S_{curr} has already branched out all its $|S_{full} - S_{curr}|$ children, it is not expandable anymore.
16: **else if** all S_{curr}'s children have been deleted **then**
17: delete S_{curr} from tree $\mathcal{T}$
18: **end if**
19: **end while**
20: Algorithm exits. If $S_{optimal}$ is null, then there is no feasible backup set that can achieve availability A_l; otherwise, $S_{optimal} - \{N_0\}$ is the optimal backup set.

server N_0. It should be clear however that the algorithm is fully distributed; every content server computes its optimal set independently. The input to the algorithm is the full set of candidate content servers (e.g., alive content servers in the pool of all content servers) $S_{candidate} = \{N_1, N_2, \ldots, N_k\}$, which practically could be obtained through some central or distributed directory services in real systems.

5.4 Optimization Algorithms for Static Configuration Problem

5.4.1 Branch-and-Bound

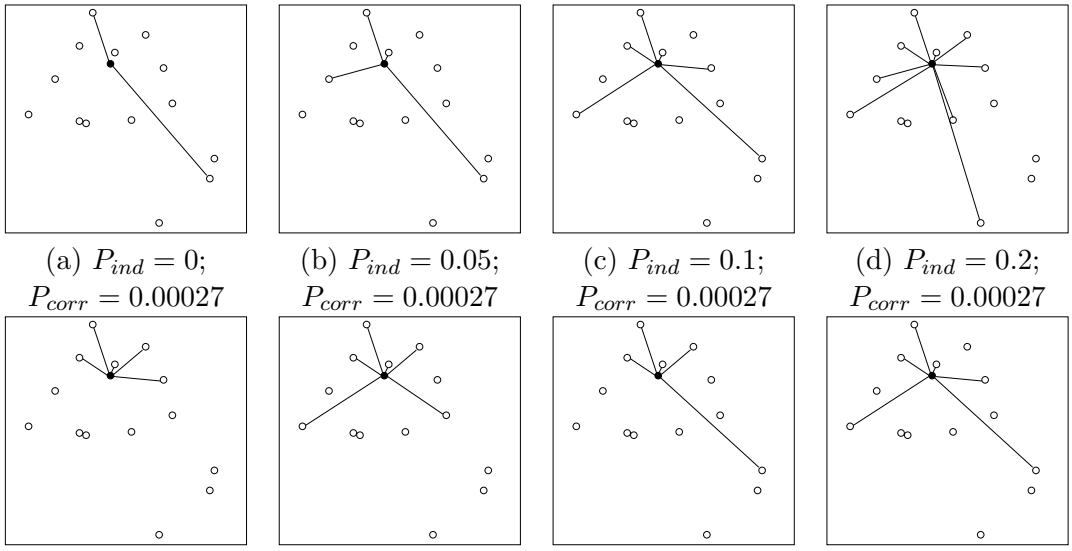
Algorithm 1 gives the pseudo-code of the branch-and-bound algorithm. The algorithm finds the minimum-cost combination of candidate nodes as the backup set for a local content server N_0. The target lower bound on the data availability is given by the user and denoted by A_l. The algorithm starts its operation from an initial set S_0, which includes the content server N_0 itself, and proceeds by incrementally expanding the backup set, adding one content

server a time. The expansion follows a tree-structure (denoted by $\mathcal{T}$), in which each tree-node represents a possible backup set. An important property of the tree is that the cost and availability of each tree-node are no less then those of any of its ancestors. This suggests that, at any stage, if the availability of a backup set satisfies the availability requirement, no further expansion of this branch is needed, since the addition of another node will result in an unnecessary increase in the cost. This allows the algorithm to keep tightening the cost upper bound (denoted by C_u) and skip/delete an entire subtree when it detects that the root of the subtree has a higher cost than the cost upper bound.

A couple of additional notes about the algorithm. First, in Algorithm 1, the tree is expanded randomly (see lines 7 and 15). Alternatively, the tree can also be expanded in a depth-first or breadth-first order, after minor changes in the code. Our experiments, however, indicate that the algorithm runs generally faster when the tree is expanded in a random order than in a depth-first or breadth-first order. Second, the initial backup set S_0 (see line 1) does not have to contain only N_0. It could comprise multiple content servers in the scenario when, for example, one or more content servers in an existing backup set of N_0 fail and one wants to keep some of the remaining content servers in the current backup set when computing the new backup set. This feature will be used in Section 5.5 when we discuss reconfiguration issues in content resiliency service networks.

5.4.2 Properties of Optimal Solutions

The branch-and-bound algorithm is practically applicable to problems with up to about 30 candidate content servers. For content resiliency service with larger number of content servers, heuristic and more efficient algorithms are called for. To draw heuristics for computationally more efficient algorithms, we do simulation-based experiments to observe the properties of optimal backup sets. We experiment with a pool of 15 content servers, which are randomly located in a circulus failure field with radius 100; the impact of fault events has an average radius of 40; the target availability lower bound is set to 0.99999; the linear and quadratic cost factors are set to 100 and 1, respectively. Fig. 28 shows the optimal backup set for a specific node with different combinations of probabilities of independent



(a) $P_{ind} = 0$; $P_{corr} = 0.00027$ (b) $P_{ind} = 0.05$; $P_{corr} = 0.00027$ (c) $P_{ind} = 0.1$; $P_{corr} = 0.00027$ (d) $P_{ind} = 0.2$; $P_{corr} = 0.00027$

(e) $P_{ind} = 0.1$; $P_{corr} = 0.00005$ (f) $P_{ind} = 0.1$; $P_{corr} = 0.00017$ (g) $P_{ind} = 0.1$; $P_{corr} = 0.0002$ (h) $P_{ind} = 0.1$; $P_{corr} = 0.00027$

Figure 28: Optimal Backup Sets

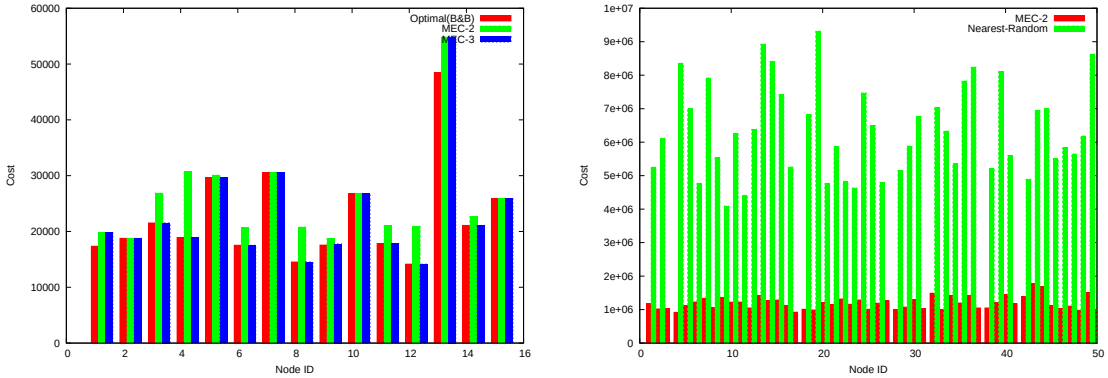
failures (P_{ind}) and correlated failures (P_{corr}). The solid nodes in the figures represent N_0 and the lines between the nodes represent the replication relationship. In Fig. 28.(a)-(d), we fix P_{corr} and observe how the the optimal backup set changes when we tune up and down P_{ind}. In Fig. 28.(e)-(h), we fix P_{ind} and observe how the the optimal backup set changes when we tune up and down P_{corr}. The figures show that the nodes in the optimal backup set can be roughly divided into two categories: the peripheral backup servers and the vicinity backup servers. The peripheral backup servers are a few (about two or three in most cases) servers that extend far away from the local server N_0. The rest of the backup set are vicinity backup servers which stay very close to N_0. Intuitively, the peripheral backup servers are chosen mainly in response to the risk of correlated failures; the farther they are from each other, the farther the backup set extends and the less likely the whole backup set fails in the same correlated fault event. However, having too much peripheral backup servers incurs much more communication cost yet does not help much in extending the backup set; the optimal set therefore chooses the rest of its members from the vicinity of N_0, mainly in response to the risk of independent failures.

5.4.3 Heuristic Algorithms Using Minimum Enclosing Circles

Algorithm 2 MEC-Preserving and Greedy

1: $S_0 = \{N_0\}$
2: $S_{candidate} = \{N_1, N_2 \ldots, N_k\}$
3: let $S_{optimal} = null$; let $S_{optimal}.cost = +\infty$
4: **for all** three-server subset $\{N_a, N_b, N_c\} \subseteq S_{candiate}$ **do**
5: find the minimum enclosing circle $\mathcal{E}$ that covers all content servers in $S_0 \cup \{N_a, N_b, N_c\}$
6: let $S_{covered}$ be the set of nodes that belong to $S_{candidate}$ and are covered by circle $\mathcal{E}$
7: **if** $(S_0 \cup S_{covered}).availability < A_l$ **then**
8: continue to next loop (goto line 4)
9: **end if**
10: let $S_{preserve}$ be the set of servers that belong to $\{N_a, N_b, N_c\}$ and are *on* the circle $\mathcal{E}$
11: let $S_{incircle} = S_{covered} - S_{preserve}$
12: Sort all servers in $S_{incircle}$ based on their distances to N_0. Assume they are $N_{i_1}, N_{i_2}, \ldots, N_{i_m}$, in decreasing order.
13: **for** $j = 1$ to m **do**
14: $S_{incircle} = S_{incircle} - N_{i_j}$
15: **if** $(S_0 \cup S_{preserve} \cup S_{incircle}).availability < A_l$ **then**
16: $S_{preserve} = S_{preserve} \cup \{N_{i_j}\}$
17: **end if**
18: **end for**
19: **if** $(S_0 \cup S_{preserve}).availability < A_l$ and $(S_0 \cup S_{preserve}).cost < S_{optimal}.cost$ **then**
20: $S_{optimal} = S_0 \cup S_{preserve}$
21: **end if**
22: **end for**
23: Algorithm exits. If $S_{optimal}$ is null, then there is no feasible backup set that can achieve availability A_l; otherwise, $S_{optimal} - \{N_0\}$ is the optimal backup set.

Based on the above observations, we propose heuristic algorithms, where minimum enclosing circles are used as a geometric indication of the extension of the backup sets. Algorithm 1 gives the pseudo-code of a heuristic algorithm that utilizes the minimum enclosing circle. The basic idea is to try out every three-server combinations of the candidate content servers — though there are exponential number of possible subsets of the candidate content servers, they share only polynomial number of minimum enclosing circles because the minimum enclosing circle of a set of content servers is determined by no more than three of the servers in the set. For each three-server combination $\{N_a, N_b, N_c\}$ under consideration, the algorithm first finds the minimum enclosing circle $\mathcal{E}$ that covers both $\{N_a, N_b, N_c\}$ and S_0 and find all candidate content servers that are covered by $\mathcal{E}$; next the algorithm preserves the servers *on* the circle and greedily deletes servers *in* the circle one by one (in decreasing order based on their distances to N_0) without violating the availability lower



(a) Heuristic versus Optimal (b) Heuristic versus Nearest-Random

Figure 29: Performance of Optimization Algorithms

bound A_l. Among all the residue sets, each computed for one three-server combination, the algorithm chooses the minimum-cost one as the final output. Same as the case in the branch-and-bound algorithm, the initial backup set S_0 does not have to contain only the local content server N_0. It could comprise multiple content servers and this is useful for the scenario that one or more servers in an existing backup set of N_0 fail and one wants to keep some of the remaining servers in the current backup set when computing the new backup set. This feature will be used in Section 5.5 when we discuss reconfiguration issues in data replication.

A variation of Algorithm 1 is to consider every two-server combinations instead of three-server combinations. In the remainder of the chapter, we refer to this variation as MEC-2 and the original one as MEC-3. Compared with MEC-3, MEC-2 considers fewer enclosing circles but it is also computationally less costly. Both run in polynomial time. Note that there are a few algorithms for computing minimum enclosing circles, the best of which has a complexity of $O(n)$, where n is the number of servers under consideration. A good summarization of these algorithms can be found in [91].

5.4.4 Performance of Heuristic Algorithms

In this section, we evaluate the performance of the heuristic algorithms. For content resiliency service networks with relative small number of servers, we are able to compare the cost of the backup sets output by the MEC-2 and MEC-3 algorithms with that of the optimal

ones output by the branch-and-bound algorithm. Fig. 29.a shows the costs of the backup sets for every servers when we use the same experiment setup described in Section 5.4.2 and set $P_{ind} = 0.1, P_{corr} = 0.00017$. The figure shows that generally MEC-3 performs better than MEC-2 but both have good performance. This indicates that the heuristics we drew by observing the properties of optimal backup sets are valid and their use in MEC-2 and MEC-3 is beneficial.

For content resiliency service networks with a large number of content servers, it is impossible for us to compare the performance of our heuristic algorithms with that of the branch-and-bound algorithm. Instead, we compare them with a nearest-random algorithm: the algorithm first determines beforehand the size of the backup set, e.g., m; then it randomly chooses m servers from a lookup scope that increases from the nearest m servers, to the nearest $2m$ servers, to the nearest $3m$ servers, ..., until it finds a solution that meets the target availability bound A_l or fails if it cannot find such a solution during the whole process. Note that the nearest-random algorithm is not one we coin for pure comparison purpose. Actually, it is a practically useful algorithm in the scenarios where extremely low computational complexity is desired.

Fig. 29.b compare the results of MEC-2 with those of nearest-random algorithm for a system with 50 nodes. The radius of the fault field is set to 1500, the average radius of fault is set to 200, and the values of other parameters are the same as in the previous experiments. The numbers for the nearest-random algorithm in the figure are the average (mean) from running the algorithm 10 times assuming the size of the backup set is 5 (we also tried other sizes, among which 5 generally gives the best backup sets in this experiment.) The figure shows that our heuristic algorithms perform significantly better than the nearest-random algorithm, largely helped by the heuristic previously learned in the case study.

5.5 Dynamic Configuration Problem in Content Resiliency Service

The problem of selecting replication locations becomes more complex when we consider the overall cost incurred in the system over a longer term (e.g., many time slots) under

dynamic system conditions. When system conditions change over time, the configuration of replication locations might need to be dynamically adjusted and this incurs additional *reconfiguration cost*. A cost-effective reconfiguration strategy for data replication should therefore consider both the static prevention cost and the reconfiguration cost, importantly, from a long-term perspective instead of a short-term point of view. Various factors could contribute to the dynamic conditions in the system. For example, the risk level of the system (e.g., the parameters in the failure models described in Section 5.3.1) could change over time. It is also possible that the recovery of failed content servers takes more than one time slots and content servers do not only fail dynamically but also come back to service dynamically. In this thesis, we concentrate on the cost effective reconfiguration problem in content resiliency networks under dynamic conditions caused by the fail-and-revive of servers. We believe that much of our results presented in this section are also applicable to systems under other dynamic conditions.

5.5.1 Problem Formulation

When a content server in the content resiliency service network fails, we assume it will be repaired and come back to service with a probability of P_{rev} during each time slot after its failure. Note that although content servers could fail both independently and correlatively, we assume their revivals are independent; intuitively, the revivals of nodes usually involve various stages (e.g., repairing hardware, powering up, restarting operating system, restarting softwares) and therefore tend to be independent from each other.

Reconfiguration Cost The reconfiguration cost is the protocol cost of changing replication configurations, e.g., setting up and tearing down the backup relationships between content servers. In this thesis, we estimate the reconfiguration cost with the number of backup relationships that need to be changed during a reconfiguration. Formally, assume the old backup set for a content server is S_{old} and the new backup set after reconfiguration is S_{new}, then we estimate the reconfiguration cost with the following equation:

$$\mathcal{R}(S_{old}, S_{new}) = |S_{old} \cup S_{new} - S_{old} \cap S_{new}| \tag{20}$$

We also assume for simplicity that all reconfigurations are conducted at the end of time

slots.

Overall Cost In a system that runs L time slots, the overall prevention cost associated with a local content server N_A includes two types of cost: the static prevention cost incurred in each time slot for replicating the data it received from local users, and the reconfiguration cost incurred during each reconfiguration of its backup set. Formally, the overall cost is

$$\mathcal{T}(N_A) = \sum_{t=1}^{L} [\mathcal{C}(N_A, S_A^t) + \beta \cdot \mathcal{R}(S_A^{t-1}, S_A^t)] \tag{21}$$

where S_A^t is the backup set of node N_A at the beginning of time slot t for $t > 0$ and $S_A^0 = \phi$. If the backup set is reconfigured at the end of time slot $t-1$, S_A^t is different from S_A^{t-1}; otherwise, they are the same. The weighting factor β is used to reflect how the actual protocol cost of setting up or tearing down one backup relationship is weighted against the static prevention cost numerically. Its value is specific to the actual method of conducting the reconfiguration.

The goal of prevention cost minimization in dynamic environments is therefore to find a reconfiguration policy that helps to decide the sequence of backup sets $\{S_A^1, S_A^2, \ldots, S_A^L\}$ such that the long-term average of the overall prevention cost (i.e., $\liminf_{L\to\infty} \mathcal{T}(N_A)$) is minimized and meanwhile the availability $\mathcal{A}(N_A \cup S_A^t)$ is maintained over the lower bound A_l for any time t.

5.5.2 Dynamic Reconfiguration Policies

A straightforward reconfiguration policy is to always recompute and reconfigure to the statically optimal backup set at the end of each time slot based on current system condition, i.e., the failed/alive state of the content servers. We refer to this policy as the *Eager-Change Policy(ECP)*. Fig. 30 shows the availability, static prevention cost (communication cost) and reconfiguration cost associated with a content server during each of 100 time slots when the ECP policy is adopted. Fig. 30.a (upper left) shows that the ECP policy maintains the availability right above the target level. Fig. 30.b (upper right) shows that the communication cost varies over different time slots, indicating that the ECP policy keeps adjusting the backup set when some content servers fail and revive over time. Figure 30.c (bottom middle) shows the reconfiguration cost incurred in each time slot. The bars below

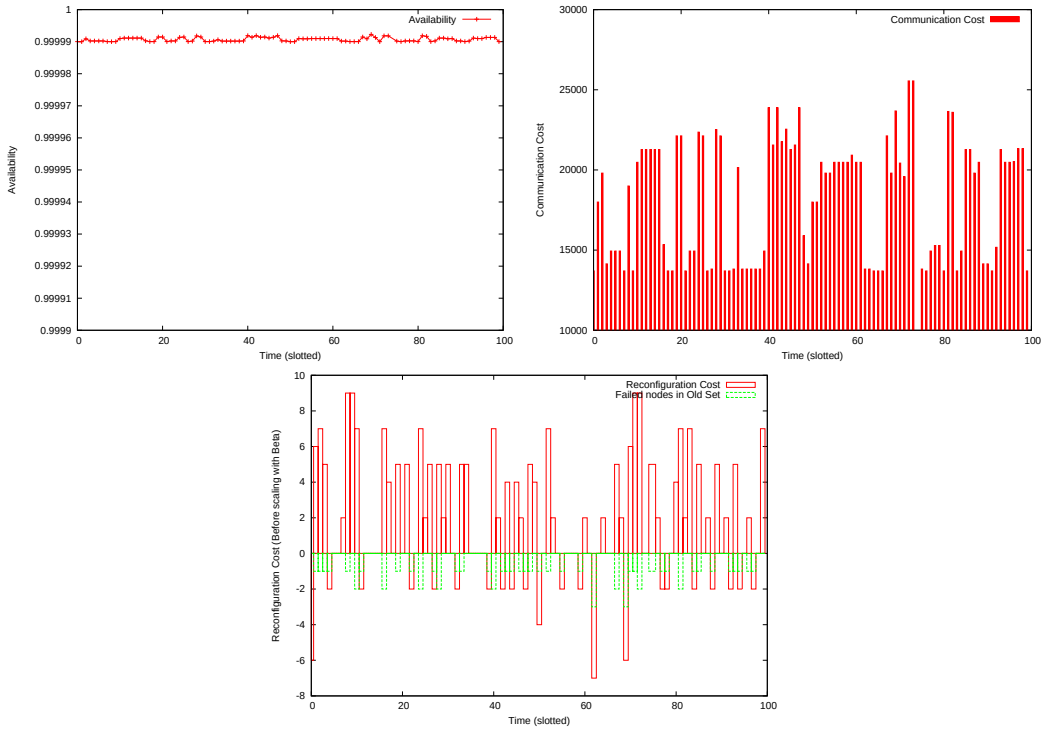


Figure 30: ECP: Availability, Communication Cost and Reconfiguration Cost

the x-axis represent the reconfigurations that happen when the availability of an old backup set drops below A_l; this happens when one more nodes in the old backup set fail during the time slot. The bars above the x-axis represent the reconfigurations that happen when the availability of old backup set is still above A_l; the reconfigurations are triggered because one or more previously failed content servers revive from failure and make better options of backup set available.

The ECP policy aggressively adapts to system conditions and therefore can minimize the total communication cost. But its aggressiveness also potentially incurs a large amount of reconfiguration cost. For systems where the reconfiguration cost is lightly weighted (i.e.,the weighting factor β in Eq. 20 is small), ECP is a good policy due to its effectiveness in reducing the total communication cost. However, for systems where the value of β is large, the reduction in communication cost can not be justified by the increase in reconfiguration cost and a less aggressive policy is more favorable. In the remainder of this section, we discuss an *Incremental Adjustment With Thresholds Policy(IATP)*, which allows us to tune

the aggressiveness of system and make tradeoffs between the reconfiguration cost and the communication cost based on different value of β.

As the name suggests, the IATP policy is incremental and threshold-based. It works as follows. At the end of each time slot t, the system checks the current backup set S_A^t for each content server N_A. There are three cases.

Case 1 No content server in S_A^t has failed in the time slot and no content server outside S_A^t has revived from previous failure either. That is, the system conditions did not change during the time slot. In this case, there is no need to adjust the backup set and therefore $S_A^{t+1} = S_A^t$.

Case 2 One or more content servers in S_A^t have failed in the time slot. In such case, the system checks the set of survived content servers in S_A^t, denoted by $\hat{S}_A^t$, and evaluate the availability of $\mathcal{A}(N_A \cup \hat{S}_A^t)$. If the availability stays above A_l, the system keeps $\hat{S}_A^t$ as the the new backup set for the next time slot, i.e., $S_A^{t+1} = \hat{S}_A^t$. Otherwise, the system needs to compute a new backup set to bring the availability back above A_l. Instead of being totally memoryless, the computation of the new backup set is incremental; it is done by *only adding* nodes to $\hat{S}_A^t$. Both the branch-and-bound and the heuristic algorithms described in Section 5.4 can provide this adding-only feature by setting the initial backup set S_0 to $\hat{S}_A^t \cup \{N_A\}$. The details have been discussed in that section.

Case 3 No content server in S_A^t has failed but one or more nodes outside S_A^t have revived from previous failures in the time slot. In this case, it is possible that a new choice of backup set with less communication cost becomes available and the system needs to make a decision on whether to reconfigure. In IATP policy, such decision is made based on a threshold as follows. The system first finds the statically optimal backup set that incurs the minimum communication cost, denoted by $\bar{S}_A^{t+1}$, based on the current set of alive candidate content servers, using the algorithms described in Section 5.4. It then calculate the ratio of the communication cost of S_A^t to that of $\bar{S}_A^{t+1}$. If the ratio is greater than a threshold ψ ($\psi \geq 1$), then the system reconfigures the backup set to $\bar{S}_A^{t+1}$, i.e., $S_A^{t+1} = \bar{S}_A^{t+1}$; otherwise, it keeps the current one, i.e., $S_A^{t+1} = S_A^t$.

The threshold ψ acts as a tunable parameter of the policy. Intuitively, the greater the

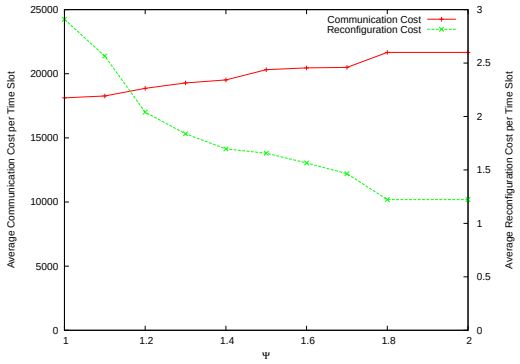
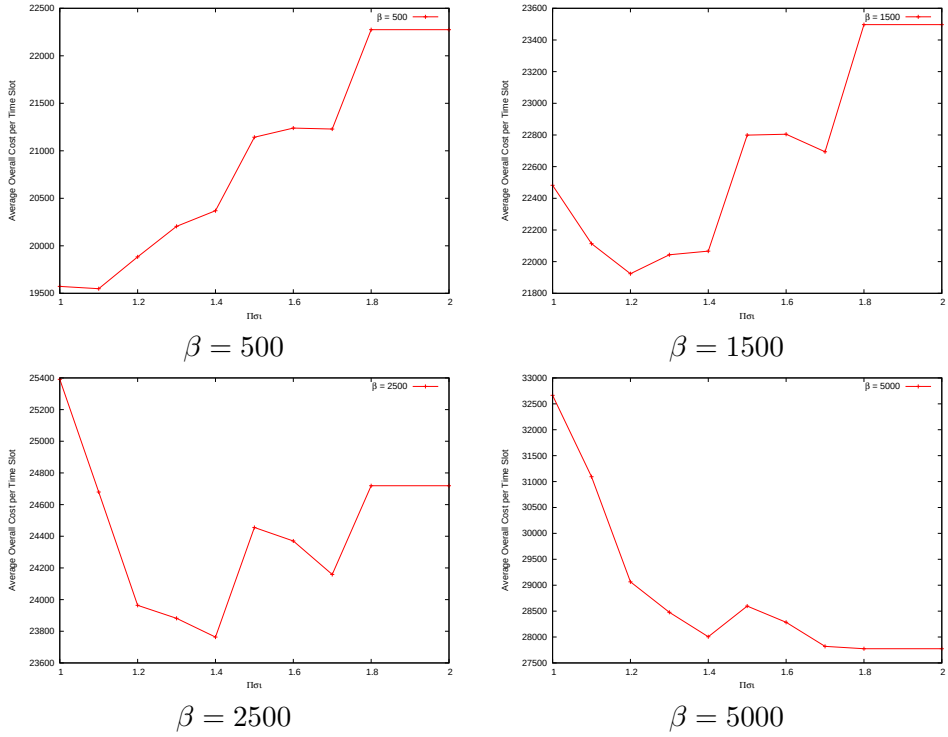


Figure 31: IATP: Communication Cost and Reconfiguration Cost Affected by Threshold Ψ

value of ψ is, the less aggressive the policy is, and this results in more communication cost but less reconfiguration cost. The choice of ψ depends on the weighting factor β. Note that ECP policy is a special case of IATP where the parameter ψ is set to 1.

Fig. 31 shows how the communication cost and reconfiguration cost are affected by the threshold Ψ when the IATP policy is adopted. The experiment parameters are the same as for Fig. 30. The figure shows that when the value of Ψ increases, the communication cost increases and the reconfiguration cost decreases. Percentage wise, the reconfiguration cost is more sensitive to Ψ than the communication cost is. This is due to the adding-only behavior of IATP when computing new backup sets in Case 2. Intuitively, according to the observations we acquired in Section 5.4, much of the backup set are vicinity nodes that stay close to N_A; reusing these nodes in the new backup set reduces the reconfiguration cost without increasing much in the communication cost. Fig.32 shows how the overall cost is affected by the threshold Ψ when the system has a different weighting factor β for the reconfiguration cost. The figures show that for small β (e.g., Fig. 32.a), ECP, which is a special case of IATP where $\Psi = 1$, is actually the best policy. This does not hold anymore for other values of β (e.g., Fig.32.b-d), and IATP provides the tunability for finding a better dynamic reconfiguration policy. The optimal value of $\Psi = 1$ depends on β and generally a greater β needs a greater Ψ.

An extension to the IATP policy is to introduce a second threshold A'_l: when the system reconfigures the backup set in Case 3, it computes the new backup set using an



$\beta = 500$ $\beta = 1500$

$\beta = 2500$ $\beta = 5000$

Figure 32: IATP: Overall Cost Affected by Threshold Ψ and Weighting Factor β

availability target A_l' instead of A_l ($A_l' \geq A_l$). This results in a policy with two tunable parameters and gives additional control on the tradeoff between the communication cost and the reconfiguration.

5.6 Summary

In this chapter, we have studied the problem of selecting cost-effective replication locations in content resiliency service networks in the presence of both correlated failures and independent failures. One critical component in the development of cost effective resiliency strategies is the assessment of failure risks for a given system. For that end, we proposed a general approach to modeling correlated failures in an representable, quantifiable and consistent way. We applied the approach to geographically correlated failures and proposed new failure models that allow for the computation of data availability in content resiliency service against both independent and geographically correlated failures. Using

the failure models, we formulated the problem of selecting cost-effective replication locations and developed two types of algorithms for solving this constrained cost minimization problem. We first developed a branch-and-bound algorithm, which is practically capable of finding real optimal solutions for content resiliency service with up to 30 servers. We then conducted case studies on small-scale problems using the branch-and-bound algorithm, observed properties of the optimal solutions, and proposed heuristic-based algorithms that are computationally efficient for large-scale problems yet generate good-quality solutions. Furthermore, for content resiliency service networks that have dynamic conditions and need dynamic reconfiguration of replication locations over time, we also investigated reconfiguration policies for reducing the long-term overall prevention cost, i.e., including both the communication cost and the reconfiguration cost. A straightforward reconfiguration policy, Eager Change Policy (ECP), was studied first and its limitations was revealed. We then proposed an Incremental Adjustment With Thresholds Policy (IATP) that provides the tunability to make tradeoffs between the communication cost and the reconfiguration cost.

Possible ways of extending the failure models, availability models and communication cost modes presented in this chapter are discussed in Section 6.2.2.

CHAPTER VI

CONTRIBUTIONS OF THE THESIS AND FUTURE WORK

In this chapter, we summarize the contributions we have made during the studies presented in this thesis. There are many directions in which these studies can be extended or supplemented. So in this chapter, we also present several such directions that we consider most interesting.

6.1 Summary of Contributions

Configuration capabilities are important for modern advanced network services. Network conditions and user populations have been significantly diversified after decades of evolution of the Internet. Configuration capabilities allow network services to be adapted to special, temporal, and managerial variations in application requirements and service operation conditions.

Network service providers need to decide on the best configuration. Ideally, a network service should have all of its components optimally configured to most effectively deliver the functionality for which it was designed. The "optimal" configuration, however, is always a compromise between different metrics. To decide on an optimal configuration, the prominent performance and cost metrics must be identified, modeled, and quantified. Optimization objective functions and constraints that combine these metrics should be formulated and optimization techniques should be developed. More important, in the scenarios where the application requirements and system conditions change over time, the service configuration needs to be dynamically adjusted and strategies that guide the reconfiguration decisions need to be developed. Because the actual process of configuring a network service incurs configuration costs, an optimal reconfiguration strategy should be one that achieves

a tradeoff between the (re)configuration costs and static optimization objectives. Furthermore, such tradeoffs must be based on the consideration of long-term benefits instead of short-term interest.

This thesis focuses on understanding the strategies for dynamic (re)configuration of advanced network services positioned above the Transport Layer. Specifically, this thesis investigates the configuration and more important dynamic reconfiguration strategies for two types of advanced network services: Service Overlay Networks, and Content Resiliency Service Networks. Unlike those network services whose configuration involves mainly arrangement of hard-wired components, these network services have the ability to change service configuration in small time scales. This makes the modeling of application requirements and system condition dynamics not only possible but also meaningful and potentially useful. Our ultimate goals in conducting the research presented in this thesis are to first develop modeling and optimization techniques for network service configuration and dynamic reconfiguration policies. We also seek to understand how effective techniques can improve the performance or reduce the cost of these advanced network services, thus demonstrating the advantage of allowing configurability in these advanced network services.

The contribution of this thesis are summarized as follows:

Configuring overlay network topology for static demands In this thesis, we studies the static topology configuration problem for service overlay networks. The thesis identifies relevant factors in different layers of a service overlay network, discussed types of cost that could be incurred in the system, and formally defined the static topology configuration problem for service overlay networks as the problem of finding the optimal overlay topology that can minimize the overall static operation cost incurred in an service overlay network for given static communication requirements. The thesis then discusses the complexity of the static overlay topology configuration problem and shows that the problem is NP-hard. The thesis also presents a simulated annealing based heuristic method of finding the optimal solution for the static topology configuration problem. The thesis conduct performance evaluation that shows the heuristic method has generate overlay topologies with good quality and is applicable to service overlay networks with large number of nodes.

The thesis also gives an integer linear programming formulation for the the static topology configuration problem for service overlay networks. Such a formulation allows general tools for solving integer linear programming problems to be applied to the static overlay topology configuration problem. This is especially useful for service overlay networks that has only small number of nodes.

Configuring overlay network topology for dynamic demands

This thesis studies the problem of dynamically reconfiguring the topology of an overlay network in response to the changes in the communication requirements. The thesis identifies the relevant factors in a service overlay network that affect the cost of using the overlay. The thesis considers two costs of using an service overlay network: the occupancy cost and the reconfiguration cost. This thesis presents the overhead involved in the reconfiguration of overlay topology by analysis as well as experiment over the Planet-Lab. The ideal goal is to find the optimal reconfiguration policies that can minimize the potential overall cost of using an overlay. The problem is NP-hard and good approximate policies are called for.

Through experiments on small systems with Markovian properties, the thesis studies the structure of the dynamic overlay topology reconfiguration problem, as well as the properties and structures of the optimal reconfiguration policies. To do so, the thesis presents a Markov decision process model for the dynamic overlay topology reconfiguration problem and solves the small Markovian cases using Howard's policy-iteration algorithm. Based on the heuristics collected from the case studies, the thesis proposes various heuristic-based methods of constructing different flavors of reconfiguration policies, i.e., never-change policy, always-change policy and cluster-based policies, to mimic and approximate the optimal ones.

The thesis evaluates the quality of the dynamic overlay topology reconfiguration policies constructed using these heuristic methods and shows that the constructed policies achieve good performance. The thesis provides evidence that the high configurability built in the overlay networks is beneficial for reducing the total operation cost when appropriate dynamic reconfiguration policies are use. The thesis also shows that these policy construction methods are helpful for studying other overlay design problems.

Configuring replication relationship in content resiliency service network

This thesis studies the problem of selecting cost-effective replication locations in content resiliency service network in the presence of both correlated failures and independent failures. One critical component in the development of cost effective resiliency strategies is the assessment of failure risks for a given system. For that end, this thesis proposes a general approach for modeling correlated failures in an representable, quantifiable and consistent way. The thesis then applies the approach to geographically correlated failures and proposes new failure models that allow for the computation of data availability in content resiliency service against both independent and geographically correlated failures.

Using the failure models, the thesis formulates the problem of selecting cost-effective replication locations and developed two types of algorithms for solving this constrained cost minimization problem. It first develops a branch-and-bound algorithm, which is practically capable of finding real optimal solutions for content resiliency service with up to 30 servers. It then conducts case studies on small-scale problems using the branch-and-bound algorithm, observes properties of the optimal solutions, and proposes heuristic-based algorithms based on minimum enclosing circles. For content resiliency service networks that have dynamic conditions and need dynamic reconfiguration of replication locations over time, the thesis also investigates dynamic reconfiguration policies for reducing the long-term overall prevention cost, i.e., including both the communication cost and the reconfiguration cost. The thesis discusses a straightforward reconfiguration policy, Eager Change Policy (ECP), and reveals its limitations. It then proposes an Incremental Adjustment With Thresholds Policy (IATP) that provides the tunability to make tradeoffs between the communication cost and the reconfiguration cost.

6.2 Future Work

6.2.1 Dynamic Configuration of Sensing Overlay Networks

A sensing overlay network is an overlay network formed over many sensing devices and processing devices. A sensing device could be any kind of instrument, including both hardware and software, that can collecting information from the environment where it resides. Examples of sensing devices include sensors developed for automatic manufacture systems,

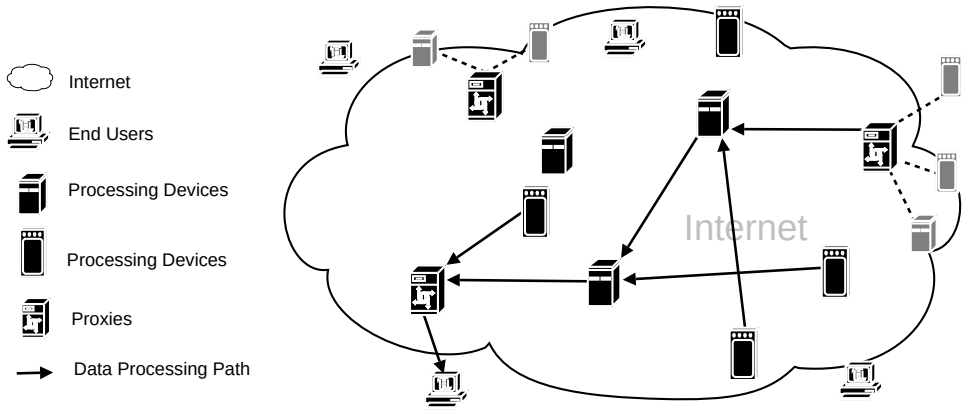


Figure 33: Sensing Overlay Network

military and meteorological satellites, radars installed at weather stations, monitoring camera installed over highways and intersections, networking monitoring software and trace collectors, and honey-pots deployed to trace computer viruses and spam. And a processing device can be any instrument that provides some specific functions of processing the information. Examples of processing devices can be as complex as a video/audio integration unit or as simple as a general purpose computer running various processing software. Note that a sensing device can also have processing capabilities and act as a processing device simultaneously. The recent rapid improvement in wireless technologies and overlay technologies has opened a door for researchers to investigate the possibility of connecting all sensing and processing devices over the Internet and integrate all their sensing/processing capabilities into a massive sensing network that supports large scale complex queries from the users.

Fig.33 presents a possible architecture for sensing overlay networks. In the figure, the sensing overlay network is formed over sensing devices, processing devices, and proxies, which act as representatives for sensing/processing devices with weak network connectivity. When an end user submit a complex query (a query that involves many information sources and multiple stages of processing) to the sensing overlay network, entities who can contribute in responding the query cooperate to form a runtime overlay topology for the query, along which the information flow from the sensing entities to the processing entities and finally to

the end user. Note that, to avoid explosion of incoming traffic at the end user, information is processed (e.g., filtered, transformed, and aggregated) while flowing in the overlay. This is a important feature that distinguishes information fusion networks from information distribution networks.

Query-specific topology configuration is an important issues in sensing overlay networks. Usually, when end users specify the information sources in a complex queries, they specify only the properties (e.g., location, type, accuracy) of the sources instead of the exact physical sensing devices. As a result, there could be multiple sensing devices that can provide the same piece of information required by the query. Similar the case is for processing elements in the query. The many-to-one relationship between the logical elements in a complex queries to the physical sensing/processing devices renders a large number of possible ways in which a query can be actually performed. Different options result in different overlay topologies, which incur different operation cost (e.g., bandwidth consumption) or performance (e.g.,delay). The problem is more complex when multiple users submit queries in the same time. Instead of forming one topology for each query, the sensing overlay network can merge the elements of all ongoing queries and form a common topology.

Dynamic configuration issues are especially interesting in sensing overlay networks because of the interactive nature of queries — users may change their queries in very short time intervals and therefore the overlay topology may need to be continually updated. This dynamic configuration problem is very similar to dynamic configuration problem we have discussed in Chapter IV in the context of service overlay networks. Essentially, we believe that the principle of aggressiveness will be the same for the dynamic reconfiguration policies in this new context. But we also expect that the problem formulation in the new context will be different enough for calling for new optimization techniques.

6.2.2 Extension to The Study of Content Resiliency Service

In this section, we discuss potential directions of extending the models, e.g., failure models, availability models and cost models, that has been presented in Chapter V. Note that such extensions change the formulation of the static configuration problem presented in

Section 5.3, and as a result, the optimization algorithms developed in Section 5.4 may not still be applicable and different optimization algorithms may need to be developed.

Failure Models: The geographically correlated failure model presented in Section 5.3.1 can be extended in at two ways.

First, the failure model presented in Section 5.3.1 assumes the distribution of fault events in the fault field is uniformly random. This uniform distribution function can be generalized to any probability function over the two-dimension fault field. For example, historic statistic data about earthquakes [41] can be used to generate such a distribution model for fault events.

Second, the failure model presented in Section 5.3.1 uses impact circles to model the impact of fault events. In the model, while the size of the circle is probabilistic, once the impact circle is given, the impact of the fault event to content servers is deterministic — a content server either fails or survives totally depending on whether it resides inside or outside the circle. A more general impact model could be one that describes the impact of fault events with a probabilistic function of the distance between the fault event and the content servers. In such case, when a fault event happens, a content server that is local to the fault event does not deterministically fails before a remote content server fails. While impact model presented in Section 5.3.1 is intuitively more suitable for modeling power outages, this more general impact model is intuitively more suitable for modeling environmental risks such as earthquake and storms.

Communication Cost Models: In Section 5.3.3 we model the communication cost of moving data from a content server to another content server with a cost function is composed of a linear term and a quadratic term of the Euclidean distance (geographical distance in this case) between the two content servers. In practice, such communication cost model is not necessarily an accurate one — while the cost of network links is intrinsically a function of the geographic distances spanned those links[83], the communication cost between two content servers may not take a simple form due to variation in network topologies, server locations and BGP routing policies. The most general communication cost model is a all-pair cost matrix that characterizes the communication cost between every pair of content servers, as

the one described in Section 3.2.1 where we discuss service overlay networks. With such a communication cost model, the branch-and-bound optimization algorithm presented in Section 5.4 still applies. The heuristic-based algorithms presented in Section 5.4 also still applies, except that "vicinity" is now defined in terms of communication cost instead of geographical distance while "peripheral" is still defined in terms of geographical distance.

Furthermore, in Section 5.3.3, we use a relative simple communication cost model by assuming data are delivered between content servers using unicast. An alternative is to have content servers relay traffic during the replication process, e.g., using overlay multicast. The communication cost formulation will become much more complex, addressing not only the communication cost between pairs of servers but also the actual relaying protocols and multicast overlay topologies. When relaying is used, expanding a backup set does not necessarily monotonically increase the communication cost for delivering data to the backup set. The branch-and-bound algorithm hence does not apply anymore because this breaks the bounding logic in the algorithm. However, the heuristic-based algorithms should still apply because they do not rely on the bounding logic. But their performance under this new context needs further investigation.

Availability Models: The availability model in the formulation of the static configuration problem will be different if erasure coding [79] is used in content resiliency service networks. Instead of straightforwardly replicating a whole trunk of data to multiple nodes and hoping at least one of them will survive the failures, systems using erasure coding transfer a data of n blocks into an encoding of m ($m > n$) blocks (in such a way that one can recover the original data from only n of the m blocks), and then distribute each of the m encoding blocks to a certain number of backup content servers, hoping at least n of m encoding blocks will survive the failures.

When erasure coding is used, a piece of data is available if n of its m encoding blocks can be collected from alive content servers. Because the calculation of availability is different, the minimization of the prevention cost depends on both the redundancy factor $\rho = \frac{m}{n}$ and the selection of m replication locations. It is interesting to make a comparison in terms of cost-effectiveness between the a content resiliency service network using straight replication

and one using erasure coding. Similar comparison has been previously conducted in the literature [10][60], but under the assumption that failures are independent only. It would be interesting to see if their conclusions still hold when correlated failures are also considered, especially when the probability of correlated failures can be quantified using a good failure model.

REFERENCES

[1] Anagnostopoulos, A., Michel, L., Hentenryck, P. V., and Vergados, Y., "A simulated annealing approach to the traveling tournament problem," in *Proceedings CPAIOR'03*, (Montreal, Canada), 2003.

[2] Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., and Morris, R., "Resilient overlay networks," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP-01)*, (New York), pp. 131–145, 2001.

[3] Awduche, D. O., Chiu, A., Elwalid, A. I., Widjaja, I., and Xiao, X., "Overview and principles of Internet traffic engineering," RFC 3272, Internet Engineering Task Force, May 2002.

[4] Bahiense, L., Barahona, F., and Porto, O., "Solving steiner tree problems in graphs with lagrangian relaxation.," *J. Comb. Optim.*, vol. 7, no. 3, pp. 259–282, 2003.

[5] Bakkaloglu, M., Wylie, J. J., Wang, C., and Ganger, G. R., "Modeling correlated failures in survivable storage systems," in *Fast Abstract at International Conference on Dependable Systems & Networks*, IEEE, June 2002.

[6] Balldine, I. and Rouskas, G., "Dynamic load balancing in broadcast WDM networks with tuning latencies," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, March/April 1998.

[7] Banerjee, S., Bhattacharjee, B., Kommareddy, C., and Varghese, G., "Scalable application layer multicast," in *Proceedings of the ACM SIGCOMM 2002*, (New York), pp. 205–220, 2002.

[8] Banerjee, S., Bhattacharjee, B., and Kommareddy, C., "Scalable application layer multicast," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 205–217, 2002.

[9] Beck, M., Moore, T., and Plank, J. S., "An end-to-end approach to globally scalable programmable networking," in *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, ACM Press, 2003.

[10] Bhagwan, R., Tati, K., Cheng, Y., Savage, S., and Voelker, G., "Total Recall: System support for automated availability management," in *Proceeding of First ACM/Usenix Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[11] Boorstyn, R. R. and Frank, H., "Large-scale network topological optimization," *IEEE Transactions on Reliability*, 1977.

[12] bram cohen, "Bittorrent protocol, http://www.bittorrent.com/protocol.html," 2003.

[13] CAIDA, "Macroscopic topology measurements project." http://www.caida.org/analysis/topology/.

[14] Calvert, K. and Zegura, E. W., "Gt-itm: Georgia tech internetwork topology models." http://www.cc.gatech.edu/fac/Ellen. Zegura/gt-itm/gt-itm.tar.gz, 1996.

[15] Calvert, K. L., Doar, M. B., and Zegura, E. W., "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, pp. 160–163, June 1997.

[16] Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., and Singh, A., "Splitstream: high-bandwidth multicast in cooperative environments," in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, (New York, NY, USA), pp. 298–313, ACM Press, 2003.

[17] Castro, M., Druschel, P., Kermarrec, A.-M., and Rowstron, A., "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communication (JSAC)*, vol. 20, Oct. 2002.

[18] Chawathe, Y., McCanne, S., and Brewer, E. A., "Rmx: Reliable multicast for heterogeneous networks.," in *INFOCOM*, pp. 795–804, 2000.

[19] Chawathe, Y. D., *Scattercast: an architecture for internet broadcast distribution as an infrastructure service*. PhD thesis, 2000. Chair-Eric A. Brewer.

[20] Chu, Y.-H., Rao, S. G., and Zhang, H., "A case for end system multicast," in *ACM SIGMETRICS 2000*, (Santa Clara, CA), pp. 1–12, ACM, June 2000.

[21] Chu, Y.-H., Rao, S. G., and Zhang, H., "A case for end system multicast," in *ACM SIGMETRICS 2000*, (Santa Clara, CA), pp. 1–12, ACM, June 2000.

[22] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W., "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001.

[23] Clip2, "The gnutella protocol specification v0.4." http://www9.limewire.com/developer/gnutella-protocol-0.4.pdf, 2002.

[24] Cugola, G., Frey, D., Murphy, A. L., and Picco, G. P., "Minimizing the reconfiguration overhead in content-based publish-subscribe," in *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pp. 1134–1140, ACM Press, 2004.

[25] Dhondy, N. and Peteren, D., "GDPS: The e-business availability solution." IBM White Paper, March 2004.

[26] Duan, Z., Zhang, Z.-L., and Hou, Y. T., "Service overlay networks: Slas, qos, and bandwidth provisioning," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 870–883, 2003.

[27] Elbaum, R. and Sidi, M., "Topological design of local-area networks using genetic algorithms," *IEEE/ACM Trans. Netw.*, vol. 4, no. 5, pp. 766–778, 1996.

[28] Eriksson, H., "Mbone: the multicast backbone," *Commun. ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[29] Faloutsos, M., Faloutsos, P., and Faloutsos, C., "On power-law relationships of the internet topology," in *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, (New York, NY, USA), pp. 251–262, ACM Press, 1999.

[30] Fan, J. and Ammar, M. H., "Dynamic topology reconfiguration of overlay networks: Structure and approximation of optimal policies," 2004.

[31] Fan, J., Pendarakis, D., and Liu, Z., "Cost-effective data resiliency: A study of geographically correlated failures and their impact on data replication strategies," 2004.

[32] Feamster, N., Balakrishnan, H., Rexford, J., Shaikh, A., and van der Merwe, J., "The case for separating routing from routers," in *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, ACM Press, 2004.

[33] Fink, R., "Network integration — boning up on IPv6 — the 6bone global test bed will become the new Internet," *Byte Magazine*, vol. 23, pp. 96NA–3–96NA–8, Mar. 1998.

[34] Francis, P., "Yoid : Extending the multicast internet architecture." White Paper, http://www.aciri.org/yoid, April 1999.

[35] Ganesan, D., Govindan, R., Shenker, S., and Estrin, D., "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pp. 251–254, 2001.

[36] Gerstel, O., Cidon, I., and Zaks, S., "The layout of virtual paths in atm networks," *IEEE/ACM Transaction of Networking*, vol. 4, no. 6, pp. 873–884, 1996.

[37] Gold, R., Gunningberg, P., and Tschudin, C., "A virtualized link layer with support for indirection," in *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pp. 28–34, ACM Press, 2004.

[38] Hadama, H., Kawamura, R., Izaki, T., and Tokizawa, I., "Direct virtual path configuration in large-scale atm networks.," in *INFOCOM*, pp. 201–207, 1994.

[39] Heckmann, O. and Bock, A., "The eDonkey 2000 Protocol," Tech. Rep. KOM-TR-08-2002, Multimedia Communications Lab, Darmstadt University of Technology, Dec. 2002.

[40] Howard, R. A., *Dynamic Programming and Markov Processes*. Cambridge, Massachusetts: The MIT Press, 1960.

[41] http://www.data.scec.org/Module/s2act04.html, "An in-depth look at earthquake distribution."

[42] http://www.kazaa.com, 2000.

[43] http://www.naptser.com, 2000.

[44] http://www.skype.com, 2004.

[45] Ingber, L., "Simulated annealing: Practice versus theory," *Mathl. Comput. Modelling*, vol. 18, no. 11, pp. 29–57, 1993.

[46] Jannotti, J., Gifford, D., Johnson, K., Kaashoek, M., and O'Toole, J., "Overcast: Reliable multicasting with an overlay network," in *the 4th Symposium on Operating Systems Design and Implementation*, 2000.

[47] Jin, C., Chen, Q., and Jamin, S., "Inet: Internet topology generator," 2000.

[48] Keromytis, A. D., Misra, V., and Rubenstein, D., "SOS: Secure overlay services," in *Proceedings of the ACM SIGCOMM 2002*, (New York), pp. 61–72, 2002.

[49] Kershenbaum, A., *Telecommunications network design algorithms*. New York, NY, USA: McGraw-Hill, Inc., 1993.

[50] Kirkpatrick, S., "Optimization by simulated annealing: quantitative studies," *Journal of Statistical Physics*, vol. 34(5/6), pp. 975–986, 1984.

[51] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[52] Krishnamurthy, B., Wills, C., and Zhang, Y., "On the use and performance of content distribution networks," in *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 169–182, ACM Press, 2001.

[53] Kwon, M. and Fahmy, S., "Topology-aware overlay networks for group communication," in *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pp. 127–136, ACM Press, 2002.

[54] Lakshminarayanan, K., Stoica, I., Balakrishnan, H., and Katz, R., "OverQoS: Offering Internet QoS Using Overlays," in *1st HotNets Workshop*, (Princeton, NJ), October 2002.

[55] Leblanc, L. J. and Narasimhan, S., "Topological expansion of metropolitan area networks," *Comput. Netw. ISDN Syst.*, vol. 26, no. 9, pp. 1235–1248, 1994.

[56] Li, Z. and Mohapatra, P., "QRON: QoS-aware routing in overlay networks," 2003. IEEE JSAC, 2003, to appear.

[57] Li, Z. and Mohaparta, P., "The impact of topology on overlay routing service," in *Proceedings of IEEE INFOCOM'04*.

[58] Li, Z. and Mohapatra, P., "Impact of topology on overlay routing service.," in *INFOCOM*, 2004.

[59] Liebeherr, J. and Beam, T. K., "Hypercast: A protocol for maintaining multicast group members in a logical hypercube topology," in *NGC '99: Proceedings of the First International COST264 Workshop on Networked Group Communication*, pp. 72–89, Springer-Verlag, 1999.

[60] Lin, W. K., Chiu, D. M., and Lee, Y. B., "Erasure code replication revisited," in *Fourth International Conference on Peer-to-Peer Computing (P2P'04)*, (Zurich, Switzerland), August 2004.

[61] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1092, 1953.

[62] Minoux, M., "Network synthesis and optimum network design problems: Models, solution methods and applications," *Networks*, no. 19, pp. 313–360, 1989.

[63] Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *ACM SIGCOMM 2001*, (San Diego, CA), September 2001.

[64] Najjar, W. and Gaudiot, J.-L., "Network resilience: A measure of network fault tolerance," *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 174–181, 1990.

[65] Nicola, V. F. and Goyal, A., "Modeling of correlated failures and community error recovery in multiversion software," *IEEE Trans. Softw. Eng.*, vol. 16, no. 3, pp. 350–359, 1990.

[66] Oram, A., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2001.

[67] Padmanabhan, V. N. and Sripanidkulchai, K., "The case for cooperative networking," in *Peer-to-Peer Systems: First International Workshop, IPTPS 2002*, (Cambridge, MA, USA), pp. 178–190, 2002.

[68] Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M., "ALMI: An application level multicast infrastructure," in *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, (San Francisco, CA, USA), pp. 49–60, Mar. 2001.

[69] Peterson, L., Anderson, T., Culler, D., and Roscoe, T., "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, 2003.

[70] Peterson, L., Shenker, S., and Turner, J., "Overcoming the Internet Impasse Through Virtualization," in *Proceedings of the 3rd ACM Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[71] Pincus, M., "A monte carlo method for the approximate solution of certain types of constrained optimization problems," *Oper. Res.*, vol. 18, pp. 1225–1228, 1970.

[72] Pioro, M. and Medhi, D., *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers (an imprint of Elsevier), 2004.

[73] Pioro, M. and Medhi, D., *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers (an imprint of Elsevier), 2004.

[74] Randy H. Katz, Peter Ming-Chien Chen, A. L. C. D. E. K. L. K. L. E. L. M. S. S. and Patterson, D. A., "Raid-ii: Design and implementation of a large scale disk array controller," Tech. Rep. UCB/CSD-92-705, EECS Department, University of California, Berkeley, 1992.

[75] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., "Topologically-aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM'02*, 6 2002.

[76] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S., "A scalable content-addressable network," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pp. 161–172, ACM Press, 2001.

[77] Ratnasamy, S., Handley, M., Karp, R. M., and Shenker, S., "Topologically-aware overlay construction and server selection.," in *INFOCOM*, 2002.

[78] Redundancy, E. R., "Appears in proceedings of the 11th ieee international conference on network protocols (icnp 2003)."

[79] Rizzo, L., "Effective erasure codes for reliable computer communication protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 2, pp. 24–36, 1997.

[80] Rouskas, G. N. and Ammar, M. H., "Dynamic reconguration in multihop WDM networks," *Journal of High Speed Networks*, 1995.

[81] Rowstron, A. and Druschel, P., "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, 2001.

[82] S. Pierre, M. A. Hyppolite, J. M. B. and Dioume, O., "Topological design of computer communication networks using simulated annealing," *Engineering application of artificial intelligence*, 1995.

[83] Shi, S. and Turner, J., "Issues in overlay multicast networks: Dynamic routing and communication cost," 2002.

[84] Shi, S. and Turner, J. S., "Routing in overlay multicast networks.," in *INFOCOM*, 2002.

[85] Sundararaj, A., Gupta, A., and Dinda, P., "Dynamic topology adaptation in virtual networks of virtual machines," in *LCR 2004: Proceedings of the Seventh Workshop on Langauges, Compilers and Run-time Support for Scalable Systems.*

[86] Touch, J., Wang, Y., Eggert, L., and Finn, G., "A virtual internet architecture," technical report, ISI, ISI-TR-2003-570, March 2003.

[87] Touch, J., "Dynamic internet overlay deployment and management using the x-bone," *Computer Networks*, vol. 36, no. 2-3, pp. 117–135, 2001.

[88] Treaster, M., "A Survey of Fault-Tolerance and Fault-Recovery Techniques in Parallel Systems," *ArXiv Computer Science e-prints*, Dec. 2005.

[89] Vieira, S. L. and Liebeherr, J., "Topology design for service overlay networks with bandwidth guarantees.," in *IWQoS*, pp. 211–220, 2004.

[90] Weatherspoon, H., Moscovitz, T., and Kubiatowicz, J., "Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems," in *21st IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, (Suita, Japan), October 2002.

[91] Xu, S., Freund, R. M., and Sun, J., "Solution methodologies for the smallest enclosing circle problem," *Comput. Optim. Appl.*, vol. 25, no. 1-3, pp. 283–292.

[92] Zegura, E. W., Calvert, K. L., and Donahoo, M. J., "A quantitative comparison of graph-based models for Internet topology," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 770–783, 1997.

[93] Zhang, B., Jamin, S., and Zhang, L., "Host multicast: A framework for delivering multicast to end users," in *IEEE INFOCOM*, June 2002.

[94] Zhao, B. Y., Kubiatowicz, J. D., and Joseph, A. D., "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," tech. rep., Berkeley, CA, USA, 2001.

[95] Zhuang, S. Q., Zhao, B. Y., Joseph, A. D., Katz, R. H., and Kubiatowicz, J. D., "Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination," in *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, (New York, NY, USA), pp. 11–20, ACM Press, 2001.

VITA

Jinliang Fan was born in Liaoning province, China. His hometown owns one the most beautiful mountains in Northeastern China, the Phoenix Mountain, which has left him a lot of precious memory of childhood. Jinliang received his Bachelor of Science degree in computer science in 1994 and his Master of Science degree in computer science in 1998, both from Peking University, China. Jinliang joined the Ph.D. program of College of Computing at the Georgia Institute of Technology in Fall 1998. His research efforts and interests include network security, privacy issues in network measurement, overlay networks, sensing networks, failure modeling and resiliency services. When he is not busy with his research work, he enjoys playing tennis, swimming, travelling, and reading history books.