Autonomic Core Network Management System

by

Ali Tizghadam

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical & Computer Engineering
University of Toronto

Copyright © 2009 by Ali Tizghadam

Abstract

Autonomic Core Network Management System

Ali Tizghadam

Doctor of Philosophy

Graduate Department of Electrical & Computer Engineering

University of Toronto

2009

This thesis presents an approach to the design and management of core networks where the packet transport is the main service and the backbone should be able to respond to unforeseen changes in network parameters in order to provide smooth and reliable service for the customers. Inspired by Darwin's seminal work describing the long-term processes in life, and with the help of graph theoretic metrics, in particular the "random-walk betweenness", we assign a survival value, the network criticality, to a communication network to quantify its robustness.

We show that the random-walk betweenness of a node (link) consists of the product of two terms, a global measure which is fixed for all the nodes (links) and a local graph measure which is in fact the weight of the node (link). The network criticality is defined as the global part of the betweenness of a node (link). We show that the network criticality is a monotone decreasing, and strictly convex function of the weight matrix of the network graph.

We argue that any communication network can be modeled as a topology that evolves based on survivability and performance requirements. The evolution should be in the direction of decreasing the network criticality, which in turn increases the network robustness. We use network criticality as the main control parameter and we propose a network management system, AutoNet, to guide the network evolution in real time. AutoNet consists of two autonomic loops, the slow loop to control the

long-term evolution of robustness throughout the whole network, and the fast loop to account for short-term performance and robustness issues. We investigate the dynamics of network criticality and we develop a convex optimization problem to minimize the network criticality. We propose a network design procedure based on the optimization problem which can be used to develop the long-term autonomic loop for AutoNet. Furthermore, we use the properties of the duality gap of the optimization problem to develop traffic engineering methods to manage the transport of packets in a network. This provides for the short-term autonomic loop of AutoNet architecture.

Network criticality can also be used to rank alternative networks based on their robustness to the unpredicted changes in network conditions. This can help find the best network structure under some pre-specified constraint to deal with robustness issues.

Dedication

To the memory of my mother, father, and brother

Acknowledgements

This thesis arose out of years of research that has been done since I came to UofT. By that time, I have worked with a great number of people whose contribution in assorted ways to the research and the making of the thesis deserved special mention. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

In the first place I would like to record my gratitude to Professor Alberto Leon-Garcia for his supervision, advice, and guidance from the very early stage of this research as well as giving me extraordinary experiences through out the work. Above all and the most needed, he provided me unflinching encouragement and support in various ways. His truly scientist intuition has made him as a constant oasis of ideas and passions in science. I am indebted to him more than he knows.

It is a pleasure to pay tribute also to Linda Espeut, particularly for her skill in handling precisely the administrative works.

I gratefully thank the members of my committee, Elvino Sousa, Eval De Lara, Ben Liang, and Indra Widjaja for their help and support.

In last few years, several members of my family left us. They were all hoping to see me graduate. I would like to address a very special thought to them. I wish they were all here to share this special moment with me.

Words fail me to express my appreciation to my wife Rushin whose dedication, love and persistent confidence in me, has taken the load off my shoulder. I owe her for being unselfishly let her intelligence, passions, and ambitions collide with mine.

Contents

List of Tables

List of Figures

Chapter 1

Motivation

Since the inception of networked communication systems, network and system management has been crucial in ensuring their proper functioning in aspects of configuration, performance, fault, security, and accounting. Today, expert human resources and complex systems are required to control and manage a plethora of networked devices and applications, ranging from small sensors to terabit routers. The explosion of the Internet and the proliferation of networked devices (peer-to-peer communications, grids, service overlay networks, sensor networks, mobile and wireless systems, etc.) creates unique challenges for the study and research in network and system management. The key community in this field has come to a consensus that future communication systems need to be autonomous, managing their own evolution, performance, fault, and security concerns without explicit user or administrator actions.

The main objective of this thesis is to develop a conceptual architecture for self-managing systems that can self-configure, self-protect, self-heal, and self-optimize. In recent years, there are many promising explorations in autonomous and adaptive system designs, bio-inspired techniques, intelligent management, and self-managing aspects for various networks and applications (e.g., wireless networks, Web services, service overlay networks, etc.). Motivated by ideas from these research works, we

establish a framework for self-optimized delivery of transport service in core networks.

1.1 Challenges in Management of Future Networks

For decades, the public communication infrastructure consisted of telephone and data networks. However, carriers are currently migrating to a consolidated single network based on IP technology called the Next-Generation Network (NGN) or Broadband Internet infrastructure. All IP packet based networking has emerged as the widely accepted vision for the future of telecom networks, covering the use of packetized IP communication for all types of fixed and mobile networks and services, including voice.

-New Applications, New Requirements The transformation of Internet into a general information network also caused the introduction of different Internet enabled applications. Some of the applications such as file-transfer and e-mail still expected reliability in communication. However, an increasing number of applications demanded timeliness in delivery of data. For example, while an e-mail application can wait for a random amount of time for delivery of messages, a telemedicine application transaction must be finished within a bounded time period. The TCP/IP suite was not ready for such time-sensitive services as it had been developed with the target of assured delivery of data. While the rapid growth of the Internet reflects the success of its best-effort service philosophy, the need for a more intelligent service delivery platform to offer guaranteed services is evident.

-Network Complexity Future networks and applications involving mobile agents, multitude of interconnected networking technologies, and sophisticated distributed applications drastically increase management complexity and demand

much higher degree of management efficiency. These new networks will provide a high-reliability, high quality Internet backbone for providing information technology services. One of the main challenges for migration to the new infrastructure is the maintenance and management of the whole network. Integration of IP and computing resources makes the problem of resource management more challenging.

-Cost of Management Current networks are operated using ad-hoc procedures implemented by expert human administrators. Change management in such an environment is human intensive, slow, and can result in unpredictable failures and inefficiencies requiring costly recovery. The overall cost of current management practices is estimated to occupy over 70% of corporate Information Technology (IT) budgets [1]. Current networks cannot be enlarged without a corresponding quadruple increase in management costs. As a result management has become the most significant barrier to scaling technology investment. The major components of management overheads are human resources costs, down-time costs, opportunity costs due to slow service deployment, and user training costs.

Management automation can significantly impact each of these components. Automated networks may be expanded without equivalent growth in payroll costs. Down-time can be reduced through automated policy enforcement, and systematic recovery. Service deployment and element configuration can be performed faster outside direct human control. Finally, user training can be strongly reduced since a large part of it involves systems management, and failure handling.

-Network Demand Fluctuations Many emerging applications for the Internet are characterized by highly variable traffic behavior over time that is difficult to predict. Classical approaches to network design rely on a model in which a

single traffic matrix is estimated. When actual traffic does not conform to such assumptions, desired bandwidth guarantees cannot be provided to the carried traffic.

Currently, Internet Service Providers (ISPs) use gross capacity over-provisioning and manual routing adaptation to avoid network congestion caused by unpredictable traffic. These lead to increased network equipment and operational costs. Development of routing infrastructures that optimize network resources while accommodating extreme traffic unpredictability in a robust and efficient manner will be one of the defining themes in the next phase of expansion of the Internet.

-Challenges in Automation of Management Process Attempts at automating network operations have so far met with limited success due to the design of existing management architectures. Current architectures assume a traditional client-server model in which element performance and status information is presented to human managers. Managers must collect and interpret this information in relation to network policy. Policy enforcement requires manual change management over distributed, heterogeneous element configuration repositories. Managers are further required to manually log and coordinate configuration updates across multiple elements due to lack of transactional configuration access mechanisms.

These architectural limitations create significant safety, scalability, and reliability challenges to automation. Several factors make the design of self-configuring networks under the current management structure challenging. Among them these are more important:

- The change propagation problem.
- The configuration policy problem.
- Resource allocation problem.

1.2 Possible Solution: Autonomic Networking

Next generation network providers will have an application-oriented architecture, where in the majority of cases connectivity will be provided by IP. While traditional network providers may become primarily providers of applications, we envision that core (backbone) IP transport networks will still play a central role. The main service in a core network is data transport. Core networks should continue delivering best effort service and at the same time they have to be able to meet different transport service requirements initiated by multiplicity of heterogeneous users.

Currently, network and service management techniques are mostly manual, and need human intervention, which leads to slow response times, high costs, and customer dissatisfaction. Management is a fundamental aspect of the promise of next generation networks and services. Managing network services is challenging, yet essential. The main requirements from a powerful network management system (NMS) are:

- Low cost of operation: The NMS must reduce the cost incurred by service providers in managing service delivery.
- Flexibility / Adaptability: The NMS must provide flexible management, easy extensibility in case new services are to be delivered, and must allow the provisioning / re-provisioning of resources to optimize service delivery.
- Responsiveness: The NMS must provide rapid response to each problem that occurs during the transport service delivery by network providers.
- Scalability: The NMS must ensure service delivery as the number of customers subscribing to the services offered increases.
- Homogeneity: The NMS must abstract heterogeneity and allow more homogeneous and integrated views of resources available.

- Fault-tolerance: The NMS must ensure service delivery regardless of problems that occur in the service provider network.

- Efficient use of resources: The NMS must allow the resources at the disposal of a service provider to be used in an optimal way for service delivery.

- Capability to handle variable demand: The NMS must ensure that the service delivery continues regardless of the demands put by customers on the service provider network.

- Service quality assurance: The NMS must maintain the service quality metrics agreed upon between customers and service providers. If the performance degrades below an acceptable threshold level, appropriate remedy actions need to be taken to ensure service delivery according to the pre-agreed service levels.

- Problem determination: The NMS must integrate data from the resources in the network, map between anomalies observed and their potential impact on service delivery, identify the cause of the problem, and execute some actions to remedy to the situation at hand.

- Policy-based management: The NMS must manage the network using high-level policies as guidelines for the management actions to be taken in order to ensure service delivery.

The requirements for a powerful network management system motivate the need to automate management of next generation networks by evolving to self-managing infrastructures, in order to ensure automated service delivery over such infrastructures. A self-managing system is characterized by four main properties:

- Self-configuring: Autonomic systems will configure themselves automatically in accordance with high-level policies and customer contracts. Self-configuring

properties mainly consist of the initial discovery and allocation of resources automatically. The system should know about the resources available, and should allocate these resources to the service instances according to an optimization criteria aiming to ensure the appropriate configuration decisions are made.

- Self-healing: Autonomic systems will detect, diagnose, and repair problems resulting from bugs or failures in the underlying resources. These problems can result from faults, congestions, or overloads. An intelligent mechanism is needed in order to remedy to the problem without human intervention.

- Self-optimizing: Autonomic systems will continuously (periodically or as needed) tune the available resources in order to optimize their use, both in terms of performance and / or cost. This optimization has to be done according to well-defined goals and objectives. The monitoring of resources is essential for this decision-making process, hence the need for a controllable and flexible measurement infrastructure.

- Self-protecting: Autonomic systems will anticipate, detect, identify, and protect against peaks in demand, malicious attacks, or cascading failures that remain uncorrected

1.3 Contribution

This thesis makes the following contributions.

1. We present AutoNet, an autonomic management and control system for transport networks. AutoNet is an architecture to automate data transport delivery in backbone networks. The thesis presents building blocks of the AutoNet with a stress on the algorithmic aspects of the blocks.

AutoNet tries to address the problems listed in section 1.1. AutoNet decreases the operation cost of the network by reducing human intervention. This in turn reduces the cost of maintenance and control of the whole network. The demand fluctuation problem is addressed by proposing appropriate traffic engineering and network planning methods. Some parts of the AutoNet architecture are proposed in [2, 3, 4].

2. We design and implement traffic engineering algorithms for AutoNet. A central block in AutoNet is the "general topology manager" which engineers the traffic of a customer and assigns appropriate virtual networks to the customers. Algorithms are designed to capture the effect of changes in the network. Link Criticality Index (LCI) and Path Criticality Index (PCI) are proposed to measure the sensitivity of a link or a path to changes in network demands and topology. We present Path Criticality Routing (PCR), an online algorithm to dynamically assign flows to least critical paths. The PCR algorithm is a heuristic based on LCI and PCI metrics.

   We validate and improve PCR by introducing a probabilistic interpretation of link and node criticality. A thorough analytical study of the properties of probabilistic link and node criticality is provided and improved versions of PCR are presented to further strengthen the effectiveness of our proposed traffic engineering method.

   PCR algorithm addresses traffic fluctuation problem by running flows in most reliable path(s). While PCR algorithm is mainly focused on intra-domain routing, it also helps alleviate the effect of inter-domain routing policies. Inter-domain routing may change the source of traffic from a border router to another one causing a change in active source-destination pairs of the network. PCR algorithm can sense this traffic shift and find new paths for flow assignment. Different

aspects of the PCR algorithm are presented in [5, 6].

3. Motivated by ideas from evolutionary science, we develop a framework to model the behavior of a network in response to traffic shifts and topology changes.

   Observations on LCI and PCI and promising results for PCR algorithm motivates an in depth analysis of the concept of criticality, which is the most important contribution of this thesis. A robustness theory of networks is introduced with the help of random-walk betweenness, a graph-theoretical metric to quantify topological load on a node or link. One major result in this part of the research is the observation that node/link criticality is independent of node/link position. This leads to the definition of a global metric, network criticality, which can characterize robustness level of a network as well as the criticality of its nodes/links. Network criticality is studied in detail. It is shown that network criticality is a strictly convex function of network weight matrix. The resulting convex optimization is investigated and the solution is used to derive autonomic traffic management algorithms as well as network planning procedures.

   The concept of network criticality permits us to address the robustness problem in a more general sense. Network criticality theory provides tools to capture the effect of changes in network conditions and proposes directions to address them appropriately by planning a robust network. A probabilistic version of PCR algorithm is derived as an application of network criticality theory to design robust routing algorithms. Some parts of our contribution in network criticality are presented in [7, 8, 9, 10, 11].

4. We develop a software package for traffic engineering algorithms proposed for AutoNet. PCR algorithm is implemented in C++ and tested for different networks. TOTEM [12], an open source software for traffic engineering, provides a platform for the management of network resources using PCR.

Chapter 2

Background

In this section we review three categories of relevant prior work: 1. Traffic Engineering (TE) proposals, mostly in the context of MPLS (Multi Protocol Label Switching); 2. literature on robustness including robust routing and robust network design; and 3. autonomic computing.

2.1 Traffic Engineering Algorithms

Traffic management consists of techniques to directly or indirectly adapt traffic to achieve certain objectives. Traffic engineering has received considerable attention during the last few years [13, 14, 15, 16]. Initially, traffic engineering was focused on developing solutions to allow large tier-1 service providers to optimize the utilization of their network. In these large networks, there are several possible paths to reach a given destination or border router. To achieve a good network utilization, the traffic should be spread evenly among all the available links. Unfortunately, this does not correspond to the way traditional IP routing protocols behave.

At the opposite extreme of large tier-1 providers, small providers and multi-homed corporate networks have different traffic engineering requirements. Their networks

have a simple topology and are frequently over-provisioned. The Tier 1 traffic engineering solutions are not really useful in such networks. For these smaller networks, the costly resource that needs to be optimized with traffic engineering is their inter-domain connectivity, i.e. the links that connect them to the rest of the Internet. These two problem extremes refer, respectively, to intra-domain [17, 18] and inter-domain [19] traffic engineering.

Intra-domain TE can be further split into IP-based TE (mainly IGP-weight optimization) and MPLS-based TE. IGP (Interior Gateway Protocol) weight optimization is defined for networks employing Shortest Path First (SPF) protocols, mainly OSPF (Open Shortest Path First) and ISIS (Intermediate System Intermediate System) [20]. IGP-weight adjustment aims at avoiding congestion by modifying link weights and hence adapting the routing scheme in the network [21].

Current SPF applications are based on default static link weights. Cisco suggests these weights to be inversely proportional to the link capacities for OSPF networks [22]. However, the performance of routing can be enhanced with an intelligent weight setting that takes the traffic demand matrix into consideration. It is also possible to extend the basic model with more complex characteristics of the problem, such as consideration of the link failures, multiple demand matrices, etc. [23]. The biggest challenge lying in the application of these extensions is the requirement for periodic weight changes under varying network conditions. Weight changes should be avoided as much as possible, since they bring instability to the network. Thus, obtaining a different weight vector for each possible scenario within the network (e.g. different demand matrices, unavailable links) is not a favorable solution. Robust optimization techniques should be developed to obtain a single weight setting that performs well for possible scenarios.

Traffic engineering based on MPLS is more promising than IP-based traffic engineering whose routing is only based on the destination prefix. The fundamental

problem with MPLS is to compute routes for the Label Switched Paths (LSPs) which will carry the traffic aggregates associated with the given Forward Equivalent Classes (FECs). Three well-known solutions are MIRA (Minimum Interference Routing) [24], PBR (Profile-based Routing) [25], and MATE (MPLS Adaptive Traffic Engineering) [26].

2.1.1 Minimum Interference Routing Algorithm (MIRA)

Minimum Interference Routing Algorithm (MIRA) [24] is one of the prestigious works in LSP routing. Unlike prior methods, MIRA considers the effect of source-destination pairs on a routing plan. A number (max-flow) is assigned to every source-destination pair , quantifying the maximum amount of traffic that can be sent through the network. MIRA is based on the idea that running traffic on some of the links may decrease the max-flow of source-destination pairs. This process is called ”interference”. MIRA tries to find the links that cause interference on a specific source-destination pair and avoids using them in the LSP construction phase. Indeed MIRA tries to build the paths in a way to minimize the interference. While the idea of ”interference” is a nice one, there are some problems with the algorithm introduced in [24]. MIRA concentrates on the effect of interference on just one source-destination pair while there are situations that some links can cause bottleneck on a cluster of node pairs. [25] investigates three benchmark networks: parking-lot (Fig. 3.6), concentrator (Fig. 3.7) and distributor (Fig. 3.8) and shows that MIRA is unable to respond to the network flow requests correctly and causes blocking for a huge number of incoming flows in these networks. Another issue with MIRA is its computational complexity compared with shortest path and widest shortest path. Finally MIRA is designed to provide bandwidth guaranteed paths for MPLS networks. It does not address any other QoS constraint in the network.

2.1.2 Profile-Based Routing (PBR)

Profile-based routing (PBR) is another proposal for routing of bandwidth guaranteed flows in MPLS networks [25]. PBR assumes that the source-destination pair and the traffic-profile between them are known. According to the PBR, a traffic profile is the aggregate bandwidth demand for a specific traffic class between a source-destination pair. The PBR has two phases. In the offline phase a multicommodity flow assignment problem (an optimization problem) is solved with the goal of routing as much commodity as possible. Each profile is considered as a separate commodity. The result of this phase is used to assign capacity from the links to each commodity. The online part will use these pre-allocated capacities to route the flows per class. In this phase PBR simply uses the shortest path algorithm and because of this the computational complexity of the online phase in PBR is less than MIRA and comparable with SP and WSP. However PBR has also some problems. Like MIRA, PBR only considers bandwidth and does not consider multi constraint QoS problem. Furthermore, in [27] the authors introduce a network called "Rainbow Topology" (Fig. 3.9) and show that the performance of PBR in this network is much worse than MIRA and WSP. The main reason is that PBR relies on whatever the offline part based on the traffic profile suggests which is not always appropriate.

2.1.3 MPLS Adaptive Traffic Engineering (MATE)

MPLS Adaptive Traffic Engineering (MATE) is a state-dependent traffic engineering mechanism to distribute network load adaptively [26] based on a quasi-static routing proposal by R. Gallager [28]. MATE assumes that several explicit LSPs have been established between an ingress and egress node in an MPLS domain using a standard protocol like RSVP-TE (Resource Reservation Protocol-Traffic Engineering extension) [29]. The goal of the ingress node is to distribute the traffic across the LSPs. MATE

is intended for traffic that does not require bandwidth reservation, for example best-effort traffic. The efficacy of any state-dependent traffic engineering scheme depends crucially on the traffic measurement process. MATE requires only the ingress and the egress nodes to participate in the measurement process of packet delay and loss and where the network does not provide bandwidth guaranteed services.

MIRA, PBR, and MATE are more efficient than the classical WSP (Widest Shortest Path) [30] and SWP (Shortest Widest Path) [31]. MPLS also allows to reroute LSPs, or change their bandwidth reservations, to make room for other more important ones [32], and provides protection/restoration methods in case of failures [33, 34, 35, 36] by setting up backup LSPs. Inter-domain TE is important economically given the high cost of inter-domain links. This problem is usually solved by configuring the BGP routers manually in a trial-and-error manner [37, 38].

2.2 Traffic Management Systems

There are some integrated network management systems proposed in literature to address traffic engineering issues faced in Internet. Further, some commercial products are developed to optimize the network performance. In the following some of these management systems are introduced.

2.2.1 RATES

Routing And Traffic Engineering Server (RATES) [39] is a software system developed at Bell Laboratories for MPLS traffic engineering and is built using centralized paradigm. RATES communicates only with the source of the route and spawns off signaling from the source to the destination for route setup. RATES views this communication as a policy decision and therefore uses Common Open Policy Service (COPS) protocol

[40]. RATES uses a relational database as its information store. RATES implements Minimum Interference Routing Algorithm (MIRA) [24] to route LSPs. It consists of the following major modules: explicit route computation, COPS server, network topology and state discovery, dispatcher, Graphical User Interface (GUI), an open Application Programming Interface (API), data repository, and a message bus connecting these modules. In summary, RATES is a well designed Traffic Engineering (TE) tool, but TE is only performed for the routing of bandwidth guaranteed LSPs in MPLS networks.

2.2.2 TEQUILA

Traffic Engineering for QUality of service in the Internet at LArge scale (TEQUILA) [41] proposes an integrated architecture and associated techniques for providing end-to-end QoS in a DiffServ-based Internet. In TEQUILA, a framework for Service Level Specification (SLS) has been produced, an integrated management and control architecture has been designed and MPLS and IP-based techniques are deployed. The TEQUILA architecture includes control, data and management planes. The management plane aspects are related to the concept of Bandwidth Broker (BB) and each Autonomous System (AS) should deploy its own BB. The BB includes components for monitoring, traffic engineering, SLA (Service Level Agreement) management and policy management. The TE subsystem is further decomposed into modules of traffic forecast, network dimensioning (planning), dynamic route management, and dynamic resource management.

The MPLS network dimensioning in TEQUILA is based on the ”hose model” [42] which is associated with one ingress and more than one egress nodes. The dynamic route management module considers: a) setting up the forwarding parameters at the ingress node so that the incoming traffic is routed to LSPs according to the bandwidth determined by network dimensioning, b) modifying the routing according to feedback received from network monitoring and c) issuing alarm to network dimensioning in

case available capacity cannot be found to accommodate new connection requests. The dynamic resource module aims at ensuring that link capacity is appropriately distributed among the PHBs (Per Hop Behavior) sharing a link, by appropriately setting buffer and scheduling parameters. TEQUILA provides a comprehensive architecture, but the algorithms and techniques to be implemented in TEQUILA have not been defined in detail, and their quantitative evaluation has not been carried out.

2.2.3 TEAM

Traffic Engineering Automated Manager (TEAM) [34] is an automated manager for management of DiffServ-based MPLS networks. This manager is a centralized authority for adaptively managing a DiffServ/MPLS domain and it is responsible for dynamic bandwidth and route management. TEAM is designed to provide an architecture that can manage large scale MPLS/DiffServ domains without any human intervention. TEAM constantly monitors the network state and reconfigures the network for efficient handling of network events. Under the umbrella of TEAM, new schemes for Label Switched Path (LSP) setup/tear-down, traffic routing, and network measurement are proposed and evaluated through simulations. Also, extensions to include Generalized MPLS (GMPLS) networks and inter-domain management are proposed. The main drawback of TEAM is that there is no guarantee to reach at the global optimum point. Further, TEAM is only applicable to (DiffServ-based) MPLS networks.

2.2.4 Commercial Products

Some tools also exist to allow content providers to optimize their outgoing traffic [43]. Earlier works on inter-domain TE include methods to select the best peering in a large network [43, 44]. Large network operators have also studied their traffic repartition and

their impact on inter-domain TE [45, 46, 47]. Several commercial network optimization toolboxes already exist, e.g. MATE (Cariden) [48], Netscope [49], TSOM (Alcatel) [50] and SP Guru (Opnet) [51].

All these traffic engineering tools are centralized and propose exact or heuristic methods. Most tools are suitable to solve what-if scenarios that allow a network operator to evaluate the impact of a change such as an IGP weight change. Beside this simulation mode, MATE also provides an IGP weight optimizer. All these tools except Netscope also support traffic engineering methods for MPLS networks, including he computation of backup paths for protection and restoration. Most tools rely on the knowledge of link loads and the existing MPLS LSPs, but MATE also provides a method to derive the traffic matrix from the link loads. The main drawbacks of the commercial tools are their lack of detailed technical public information about their algorithms and the difficulty to upgrade them by new research proposals.

This thesis present a method to engineer the traffic of a network based on the importance of its links and nodes. We assign a survival value to each link/node and quantify the criticality of a specific path based on the survival value of its links. The paths with low criticality are good candidates to run the demand flow. We compare our proposed traffic engineering method with some other available routing algorithms. We show that our approach is more robust and can support more diverse range of network topologies through simulations on some well-known network topologies.

2.3 Robustness in Networks

In this thesis robustness is studied from two different points of view. First, we consider design of robust networks which involves selecting appropriate topologies and assigning suitable capacities in order to provide specified degree of robustness. Second, we

consider design of on-line algorithms for robust traffic engineering in networks. In this section we highlight some previous research work in this field.

2.3.1 Robust Network Design

In [52] robustness of network topologies is studied. Graph-theoretic concepts are used to investigate which network topologies are the most robust. Authors argue that "node connectivity" is the most useful metric in graph theory to study the robustness problem. They examine the relationship between node connectivity and the degree of symmetry of the network and they suggest that it is important for robust networks to satisfy *node similarity* and *optimal connectivity* conditions. Two nodes are similar if there is an automorphism that can map one to the other. A network is node similar if all of its nodes are similar. A graph is optimally connected if its node connectivity is equal to the link connectivity metric and both equal to the minimum node degree of the graph. [52] investigates the relationship between these conditions, and arrives at the result that a network provides maximum resistance to node destruction if it is both node-similar and optimally connected. The paper then describes a number of ways to design robust networks satisfying these conditions.

[53] introduces a new measure of symmetry, symmetry ratio of a network. This metric is defined to be the ratio of the number of distinct eigenvalues of the network to the diameter. This metric is used to study the robustness of a network topology in the face of targeted attacks.

A way to design backbone networks is proposed in [54] that is insensitive to the traffic matrix (i.e., that works equally well for all valid traffic matrices), and that continues to provide guaranteed performance under a user-defined number of link and router failures. The authors use Valiant Load-Balancing method and argue that it is a promising way to design robust backbone networks. The approach was first proposed by Valiant for processor interconnection networks [55], and has received recent interest

for scalable routers with performance guarantees [56, 57]. [54] applies Valiant method to backbone network design problem and provides appropriate capacity allocation for the links of a logical full mesh topology to support load-balancing for all possible traffic matrices. In Valiant load-balancing method, traffic destined for a sink d is forwarded to intermediate hops with equal splits to all nodes, and then it is forwarded to the destination d. Delay propagation is one of the shortcomings of this method.

2.3.2 Robust Routing

Network operators would like their network to support current and future traffic matrices, even when links and routers fail. Not surprisingly, no backbone network can do this today: It is hard to accurately measure the current matrix, and harder still to predict future ones. Even if the matrices are known, how do we know a network will support them, particularly under failures? As a result, today's networks are designed in a somewhat ad-hoc fashion, using rules-of-thumb and crude estimates of current and future traffic. To tackle this problem, an abundance of work has been done.

2.3.2.1 Robust Shortest Path Problem

The Absolute Robust Shortest Path problem consists in finding a path corresponding to the minimum weight over a set of scenarios. Each scenario corresponds to a predetermined set of edge weights. The motivation for studying this problem comes from telecommunications where a communication network is used to send packets from a given source to a given destination. The aim is to determine a shortest path between some given source and destination under some criteria (total delay, congestion, . . .). The network manager has to choose a robust path where the total delay is acceptable regardless of the realized congestion [58, 59].

2.3.2.2 Robust Solution to Minimize the Maximum Regret

One common approach to robustness consists in finding a robust solution which minimizes the maximum regret, i.e., the worst case scenario. Several approaches have been proposed. In [60] the focus is on minmax regret for optimization over uniform matroid. [61] and [62] work on ellipsoidal uncertainty. [63] investigates robust optimization with control of the conservation of a solution. The focus of [64] is on minmax regret robust optimization and [65] also works on absolute and relative robustness for spanning tree problem. In Robust Shortest Path problem, the uncertainty is only related to link weights and does not take into account the possible evolutions of the network topology. Future telecommunication networks may become very dynamic, thus, an existing route between two protagonists may appear less profitable after adding new connections or nodes, and it may have to be reconsidered with the network evolution. Further cost may appear during this evolution, especially for resource dis-allocation and re-allocation.

2.3.2.3 Oblivious Routing

Another category of algorithms in the area of robust routing is oblivious routing [66, 67, 68, 69, 70, 71]. In oblivious routing, routes are computed to optimize the worst-case performance over all traffic demands, therefore, the computed routes are prepared for dynamic changes in traffic demands. In their pioneering work [66], Applegate and Cohen propose an efficient algorithm to compute the worst case oblivious routing for real networks. They also extend oblivious routing to compute failure scenarios [67]. They found that the oblivious ratio is typically around a factor of 2. A penalty as high as 100% may be acceptable when traffic demands are completely unpredictable, but it is a high cost to pay under predictable demands. In other words, oblivious routing takes a pessimistic point of view and may not be appropriate in relatively stable periods or stable networks.

There are also recent studies on the interaction of intra-domain traffic engineering with inter-domain routes and traffic. Examples include evaluation [72, 73, 74, 75, 76] and design [77, 78]. Recently, researchers observed that intra-domain traffic engineering within an AS can cause substantial traffic changes outside the AS [79, 77, 73]. For example, Agarwal et al. report in [79] that for an operational tier-1 ISP, intra-domain traffic engineering can cause up to 25% of its traffic to a neighboring AS to shift the exit point. Such traffic changes could trigger routing changes at the neighboring AS, and result in network instability. Motivated by these studies, another proposal for traffic engineering is introduced in [15]. Both prediction-based routing and oblivious routing are used in this work. Routing is optimized for predicted demands to achieve high efficiency under normal network conditions; in the meantime bounds we placed on the worst-case performance penalty to ensure acceptable performance when the network experiences unpredictable changes. As many of other research works in this category, [15] does not provide absolute bandwidth guarantee. In addition, it is an off-line traffic engineering algorithm.

An extension to Valiant load-balancing method for routing ([54]), two-phase routing, is proposed in [70, 80],where a message for a specific destination is sent to some intermediate nodes with possibly unequal split ratios and the proposed algorithm considers the problem of minimum bandwidth (physical) routing under router node capacity constraints. Worst case efficiency in two-phase routing is poor and delay propagation could not be controlled.

In this thesis we investigate the problem of oblivious routing from a different perspective. We assign a value to each path in a network to quantify its sensitivity to traffic shifts and topology changes. We propose a routing algorithm that assigns demands to less sensitive paths.

2.4 Autonomic Computing

Autonomic computing is the ability of an IT infrastructure to adapt to change in accordance with business policies and objectives. Quite simply, it is about freeing IT professionals to focus on higher-value tasks by making technology work smarter, with business rules guiding systems to be self-configuring, self-healing, self-optimizing, and self-protecting.

2.4.1 What is Autonomic Computing?

The term "autonomic" comes from an analogy to the autonomic central nervous system in the human body, which adjusts to many situations automatically without any external help. We walk up a flight of stairs and our heart rate increases. If it is hot, we perspire. If it is cold, we shiver. We do not tell ourselves to do these things, they just happen. Similarly, the way to handle the problem of managing a complex IT infrastructure is to create computer systems and software that can respond to changes in the IT (and ultimately, the business) environment, so the systems can adapt, heal and protect themselves.

The cost of technology continues to decrease, yet overall IT costs do not. With the expensive challenges that many companies face, IT managers are looking for ways to improve the return on investment of IT by:

- Reducing total cost of ownership
- Improving quality of service
- Accelerating time to value
- Managing IT complexity

The autonomic computing vision is for intelligent, open systems that:

- Manage complexity
- ”Know” themselves
- Continuously tune themselves
- Adapt to unpredictable conditions
- Prevent and recover from failures
- Provide a secure environment

Autonomic computing systems consist of four attributes. As illustrated in Fig. 2.1, they are:

-Self-configuring With the ability to dynamically configure itself, an IT environment can adapt immediately with minimal intervention to the deployment of new components or changes in the IT environment. Depending on a particular task's context, a system shall achieve that its components automatically assemble into a constellation that is both flexible and without redundancies to perform that task.

-Self-healing Autonomic systems should be conceived with the capabilities to autonomously detect, diagnose and repair localized problems resulting from software or hardware failures. In a problem situation, a system component will be attributed the task to report the bug, as well as the source to a manager component. This requires the capability of the system components to be aware of each other and each others resources. To fix a bug occurring in a system component, this component may be able to localize the appropriate manager component capable to relay the faulty component or to provide prophylaxis for healing the diseased component.

-Self-optimizing Self-optimization refers to the ability of the IT environment to efficiently maximize resource allocation and utilization to meet end users' needs

Figure 2.1: Main Attributes of Autonomic Computing

with minimal intervention. In the near term, self-optimization primarily addresses the complexity of managing system performance. In the long term, self-optimizing components may learn from experience and automatically and pro-actively tune themselves in the context of an overall business objective.

-Self-protecting A self-protecting environment allows authorized people to access the right data at the right time and can take appropriate actions automatically to make itself less vulnerable to attacks on its run-time infrastructure and business data. A self-protecting IT environment can detect hostile or intrusive behavior as it occurs and take autonomous actions to make itself less vulnerable to unauthorized access and use, viruses, denial-of-service attacks, and general failures.

2.4.2 Autonomic Computing Concepts

In an autonomic environment, components work together, communicating with each other and with high-level management tools. They can manage or control themselves and each other. Components can manage themselves to some extent, but from an overall system standpoint, some decisions need to be made by higher level components that can make the appropriate trade-offs based on policies that are in place. Let us start by looking at how a single entity is managed in an autonomic environment. Fig. 2.2 represents the control loop that is the core of the autonomic architecture.

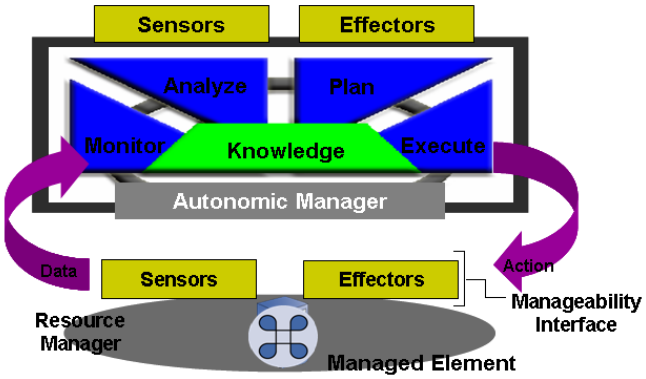


Figure 2.2: General Autonomic Control Loop

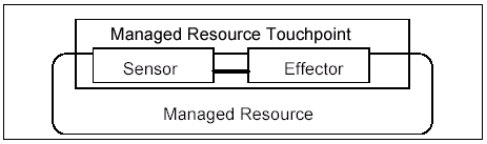


Figure 2.3: Manageability Interface in Autonomic Architecture

2.4.3 Managed Resources

The managed resource is a controlled system component. The managed resource can be a single resource or a collection of resources. The managed resource is controlled through its sensors and effectors. The sensors provide mechanisms to collect information about the state and state transitions of an element. Sensors can be implemented using a set of get operations to retrieve information about the current state, or a set of management events (unsolicited, asynchronous messages, or notifications) that flow when the state of the element changes in a significant way, or both. The effectors are mechanisms that change the state (configuration) of an element. In other words, the effectors are a collection of set commands or application programming interfaces (APIs) that change the configuration of the managed resource in some way.

The combination of sensors and effectors form the manageability interface (referred to as the touch-point; see Fig. 2.3) that is available to an autonomic manager. The

architecture encourages the idea that sensors and effectors are linked together. For example, a configuration change that occurs through effectors should be reflected as a configuration change notification through the sensor interface.

Web services can (and will) be used to implement sensor-effector functions. By utilizing a Web services architecture for communication to the managed resource touch-point, current approaches to resource management can be reused and wrapped with a Web service.

2.4.4 Autonomic Managers

The autonomic manager is a component that implements the control loop. The architecture dissects the loop into four parts that share knowledge:

- The monitor part provides the mechanisms that collect, aggregate, filter, manage, and report details (metrics and topologies) collected from an element.
- The analyze part provides the mechanisms that correlate and model complex situations. These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations.
- The plan part provides the mechanisms that structure the action needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
- The execute part provides the mechanisms that control the execution of a plan with considerations for on-the-fly updates.

The four parts work together to provide the control loop function. Fig. 2.2 shows a structural arrangement of the parts. The bold line that connects the four parts should be thought of as a common messaging bus rather than a strict control flow. In other words, there can be situations where the plan part may ask the monitor part

to collect more or less information. There could also be situations where the monitor part may trigger the plan part to create a new plan. The four parts collaborate using asynchronous communication techniques, like a messaging bus.

Chapter 3

Path Criticality Routing

In this chapter we present the conceptual idea of our management system, AutoNet. We propose a two-loop control and resource management system in AutoNet. Our focus is in one block, the general topology manager, which is the main block dealing with traffic engineering issues. The main goal of the topology manager block is to distribute traffic flows among different paths so as to maximize network robustness. We develop an algorithmic framework for robust traffic engineering methods that can be used in this block. We propose Path Criticality Routing (PCR) algorithm as a heuristic to find robust routes in AutoNet. PCR became the main inspiration for the theoretical study which follows in subsequent chapters.

3.1 The Proposed Management Approach

The conceptual idea underlying our management architecture is inspired by the concept of **survival value** in the theory of evolution. Evolutionary processes are good examples of self-organizing systems. Darwin's seminal work describing the long-term processes in life, and the theory of "natural selection" [81], suggests viewing the management architecture as an evolutionary process that evolves by natural selection

in response to environmental changes. According to the natural selection process slight variations, if useful, are preserved. Darwin's theory assigns a survival value to all the processes in the world. This value is a measure of resistance or robustness of a process or element to the changes in nature. In other words, survival value indicates how adaptable a system is to unexpected events.

In this thesis we are trying to find appropriate metrics to model and quantify the survival value of communication networks based on their characteristics including topology, capacity and offered/accepted traffic. At the same time we build conceptual architecture for autonomic management and control systems which are capable of reacting to the survival value.

Darwin's theory does not consider any "final target" for the evolutionary changes in the nature, but one can see that the survival as the goal can lead to an implicit optimization problem. Therefore we arrive at the view that the first goal of the management system is to keep the system "alive" under unforeseen circumstances. For our purposes, the system (network) can be modeled as a graph, and our main service is data transfer.

In any network, from small designed networks to large scale social networks or the Internet, connectivity is a crucial factor as it is necessary for communication purposes. Therefore, the first parameter to consider in the survival value is the connectivity of the graph. Any communication network should evolve in a way that guarantees future connectivity to the extent possible.

The management system can be viewed as an optimization process that attempts to converge to some optimum steady state that provides some required level of connectivity. Therefore, the optimization deals with the real-time efficiency and performance of the whole network as a short-term goal, while it strives to maintain and improve the survival value of the network as a long-term goal. To achieve long-term and short-term goals, one needs to have control mechanisms to monitor the present situation

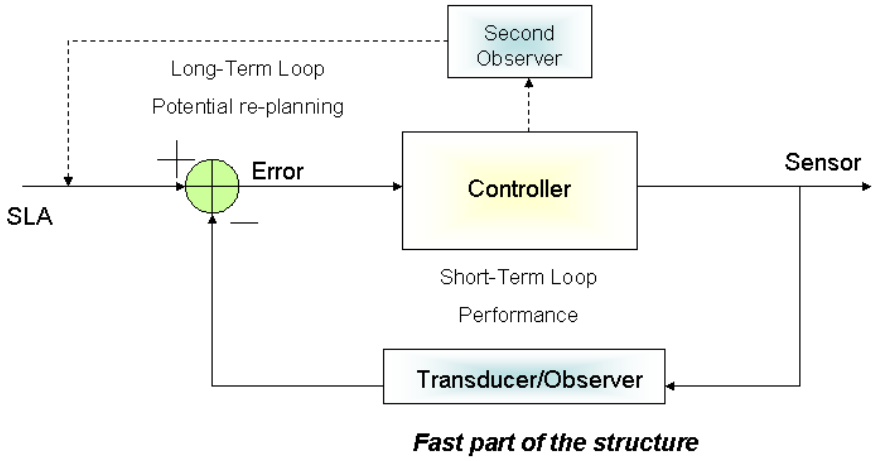


Figure 3.1: Self-Management Architecture

and make decisions accordingly using a corresponding controller. A simple model of such a system with two feedback loops is shown in Fig. 3.1. In long-term loop the control system learns a policy evolved as the result of gradual changes in the controlled system. In this evolutionary process, sometimes the control system cannot provide an appropriate policy. In this situation a re-planning process takes place and the new plan is put in place.

We used this short/long loop approach to the network management problem to design our management system, AutoNet. The main idea behind AutoNet, is to design appropriate long-term and short-term control loops to achieve the desired connectivity and performance simultaneously. Our performance metric should directly reflect our traffic engineering goal, the optimization of robustness. In particular AutoNet must provide robustness against changes in topology, such as link/node failures or changes in capacity, uncertainty in traffic demands, and changes in community of interest (active source-sink pairs). In later sections of this chapter we shed more light on this view of robustness and show that we can reach at an appropriate metric to quantify survival value by analyzing robustness properties of a network. In AutoNet, the

short-term and long-term control loops are designed to achieve robustness first and goodness or efficiency of the other QoS metrics next.

We have used IBM's "Autonomic Computing" [82] framework as a reference model and built our "Autonomic Networking" architecture based on the model's philosophy and the techniques adopted from autonomic computing (AC). While AC is useful to describe the evolution of a network to the stable position and configuration, there is still a problem with large networks such as Internet. Having the information of the Internet topology and its other attributes is next to impossible. The traffic cannot be monitored in real-time and there are unavoidable delays in procuring traffic information that is to be used to control and manage the whole network.

Furthermore, in the Internet performance is sacrificed in exchange for flexibility, scalability and robustness. In a holistic view, this makes the management of large networks a challenging task that requires human intervention in many situations. For this reason, there has been a proliferation of overlay networks on top of IP networks. Overlays and virtual networks are smaller in size and can be easily managed. It is possible to offer QoS-based services on top of overlay networks that improve on the best-effort nature of the Internet. AutoNet uses the concept of virtual networks (VN) and overlays to ease the structure and algorithms. In chapter 6, we present the architecture of AutoNet based on this methodology.

3.1.1 Conceptual Architecture of AutoNet

The overall conceptual architecture of the AutoNet autonomic management system is shown in Fig. 3.2. We introduce a hierarchical arrangement of two parts, a long-term (slow) and short-term (fast) part. The short-term part reacts to the network changes in real-time and the 'slow' part takes actions over a longer time-horizon.

The long-term part of AutoNet develops the evolution part based on an initial knowledge base that consists of the business policy as well as empirical results from

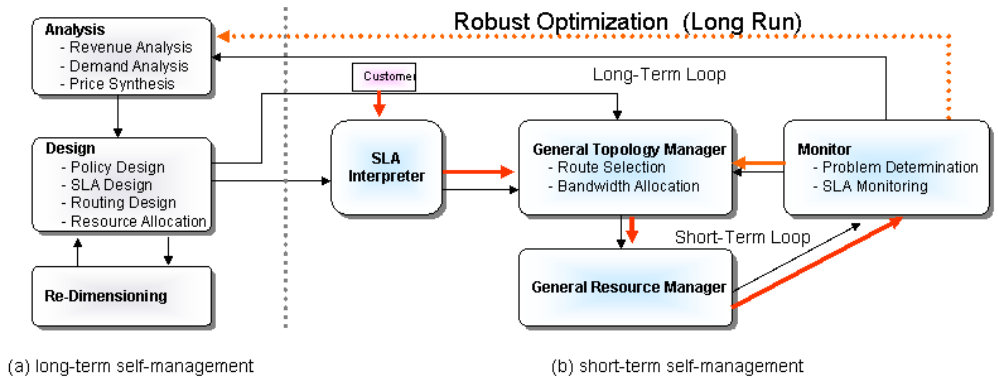


Figure 3.2: Conceptual Architecture of AutoNet

previous experience about customer demand, network element reliability, price elasticity, etc.

The network plan includes the translation of business policy into policies that are meaningful to the short-term part for use in the handling of customer requests. Our methodology is to convert the SLA to metrics from graph theory to capture both the topological aspects and SLA requirements. The plan also includes the synthesis of the SLA templates that will be offered to customers taking into account forecasted demand, resource requirements and price elasticity. All these planning parts are aimed at providing robustness through the long loop while providing immediate performance with the short-term loop. Finally, the plan also includes pre-partitioning of network resources to facilitate the handling of customer requests by the short-term part. For example, the plan may include pre-provisioned routes per each (ingress, egress) pair or more generally pre-provisioned VNs.

Because of the autonomic nature of the overall system, the short-term part needs to interact with the slow (long-term) part when carrying out certain self-healing, self-optimizing, and self-configuring functions (bringing robustness as the final result). This mainly occurs when un-predictable events take place, such as sudden surges in demand or major failures in the network, or in the case of adapting to gradual changes.

In these situations the short-term part will respond to provide a fast real-time cure, but will act to provide a long-lasting cure by making a request for re-dimensioning to the slow part, if necessary. The interaction between slow and fast parts of the system could also be the result of detecting inefficiency in resource usage in the fast part. In this case a request for re-dimensioning is sent to the slow part to re-optimize the allocation of resources. Indeed these interactions are evolutionary processes which bring stability and robustness for AutoNet.

The short-term or 'fast' part of the system consists of four major building blocks that are driven by customer requests. As shown in Fig. 3.2, the 'SLA Interpreter' block is responsible for negotiating the SLA with the customer and for converting the SLA contract to an appropriate form understandable by a 'General Topology Manager' block. This latter block plans the route (or VN) and resource allocation based on the converted SLA, the already allocated resources, and current network demands. The results are delivered to the 'General Resource Manager' block which executes orders that allocate the appropriate amount of resources. The 'Monitoring' block continuously monitors the system to identify possible problems (e.g., SLA violations, failure alarms and so on). After filtering, it sends information to the 'General Topology Manager' to develop an immediate cure, and in parallel it may send a message to the 'Analysis' block of the 'slow' part to report an unpredictable event. If appropriate, the 'Analyze' block may initiate new network planning.

We will return to this conceptual architecture in chapter 6 and we will develop the details of our system to realize this conceptual architecture. In the remaining of this chapter our focus is in the general topology manager (Fig. 3.2), which is the main block dealing with traffic engineering issues in AutoNet. We develop a fast algorithm for distributing the traffic demands to appropriate paths of the network to maximize the robustness.

3.2 Robustness Quantification

This first step towards designing a robust routing algorithm is to clarify the meaning of robustness. In this thesis we provide a quantitative definition of robustness.

3.2.1 Definition of Robustness

From a control-theoretical point of view, robustness is the capability of a network to keep itself in a stable mode when changes take place in different parameters of the network in unpredictable fashion. In order to fix our notion of robustness we begin with our definition of robustness. There are three major types of changes that may affect the performance of the network:

1. Changes in network topology including capacity.

2. Changes in community of interest (CoI), the set of active source-destination pairs.

3. Changes in Traffic demand.

Throughout this thesis, we call a ”network topology”, ”network control strategy” or a ”traffic engineering method” robust if its performance is not sensitive to changes in topology, traffic or community of interest. We aim to develop robust methods by studying the interaction between flow assignment and network structure with the help of graph-theoretical concepts.

In order to have a robust routing plan we need to recognize the effect of link and node changes on network connectivity. Connectivity is a well studied subject in graph theory [52, 53] which allows us to define some useful metrics to measure the sensitivity of the network to node or link failures.

Capacity of a network is another key issue in flow assignment problem. Clearly the paths with more capacity are desired since the low capacity paths are prone to

congestion. Hence an intelligent routing plan should avoid routing the flows onto the low capacity paths and should request for capacity increases for those paths if possible.

Finally traffic demand directly affects the routing plan. The traffic demand profile may change from time to time (e.g. week-day traffic profile). We need to find routing schemes that are robust to the predicted traffic patterns and unpredicted ones to the extent possible.

We will next introduce two metrics to estimate the effect of these characteristics: **Link Criticality Index** (LCI) and **Path Criticality Index** (PCI) which are based on the theory of graphs [83, 84]. We will subsequently propose our first heuristic routing algorithm based on PCI. This algorithm provides the "general topology manager" block of AutoNet.

3.2.2 Link Criticality Index (LCI)

Freeman [83] introduced a useful measure in graph theory called **betweenness centrality**. Suppose that we are measuring the centrality of node k. The betweenness centrality of the node is defined as the share of times a node i traverses a node k in order to reach a node j via the shortest path(s). A similar definition of betweenness centrality is also valid for the links of a network graph. Suppose $n_{sd}^{(sp)}$ is the number of shortest paths between source-destination pair $s-d$ and $n_{sd}^{(sp)}(l)$ is the number of shortest paths between source s and destination d containing the specific link l . According to the original definition of betweenness, the betweenness of link l for source-destination pair $s-d$ would be $\frac{n_{sd}^{(sp)}(l)}{n_{sd}^{(sp)}}$. The *total betweenness* of link l is the sum of shortest path betweennesses of link l for all possible source-destination pairs: $b_l = \sum_{s,d} \frac{n_{sd}^{(sp)}(l)}{n_{sd}^{(sp)}}$.

Example 3.2.1 *Fig. 3.3 shows a high-level view of a network with 3 different active source destination pairs $s_1-d_1,\ s_2-d_2,\ s_3-d_3$. There are 3 paths between s_1 and d_1 ($P_1,\ P_2,\ P_3$), 2 paths between s_2 and d_2 ($P_4,\ P_5$), and 3 paths between s_3 and d_3 ($P_6,\ P_7,\ P_8$). Among these*

paths P_1, P_3, P_5, P_6, P_7, P_8 are shortest paths. Further, paths P_1, P_2, P_5, P_7, P_8 include link l. These information are summarized in table 3.1. Based on table 3.1 shortest-path betweenness of link l can be obtained for different source-destination pairs as follows:

$$\begin{aligned} b^{(sp)}_{s_1 d_1}(l) &= \frac{1}{2} \\ b^{(sp)}_{s_2 d_2}(l) &= \frac{1}{1} \\ b^{(sp)}_{s_3 d_3}(l) &= \frac{2}{3} \end{aligned}$$

The shortest-path link betweenness is then:

$$b^{(sp)}(l) = \frac{1}{2} + \frac{1}{1} + \frac{2}{3} = \frac{13}{6}$$

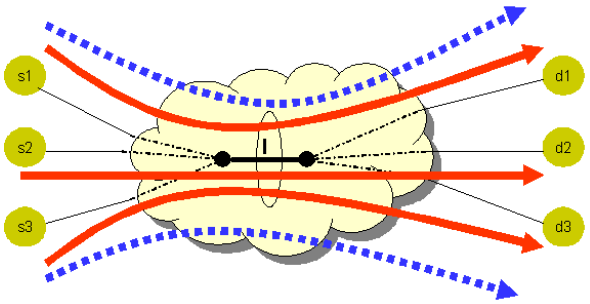


Figure 3.3: Test Network to Study Link Betweenness

Path	Src	Dest	SP?	l in Path?
P1	s1	d1	Y	Y
P2	s1	d1	N	Y
P3	s1	d1	Y	N
P4	s2	d2	N	N
P5	s2	d2	Y	Y
P6	s3	d3	Y	N
P7	s3	d3	Y	Y
P8	s3	d3	Y	Y

Table 3.1: Path Characteristics for the Test Network

In order to provide robustness, the shortest path is not necessarily the best path because it may overload some paths and underload other possible paths. For this

reason we modified the definition of link/node betweenness (we call it deterministic betweenness) as follows:

Definition 3.2.2 *Let n_{sd} be the total number of simple (loop-free) paths between source-destination pair $s-d$ and $n_{sd}(l)$ be the total number of simple paths between source s and destination d containing the specific link l. Now one can define deterministic betweenness of link l for source destination pair $s-d$ as the fraction $\frac{n_{sd}(l)}{n_{sd}}$. The total deterministic betweenness of link l is the sum of all these fractions for active source-destination pairs.*

$$b(l) = \sum_{(s,d)\in CoI(G)} \frac{n_{sd}(l)}{n_{sd}} \tag{3.1}$$

By active source-destination pairs in definition 3.2.2 we mean those nodes which are actively sending and/or receiving traffic,or in brief community of interest for network G (CoI(G)). One can easily see two major differences between our modified definition of betweenness and the original shortest path one:

- In deterministic betweenness all the paths are involved, whereas in shortest-path betweenness only shortest paths are considered.
- In deterministic betweenness only the active path set (CoI) is involved in definition of betweenness, whereas in original shortest path betweenness, all possible node pairs are considered.

Example 3.2.3 *For the network of example 3.2.1, one can find the deterministic betweenness of link l for different source-destination pairs as follows:*

$$\begin{aligned} b_{s_1ld_1} &= \frac{2}{3} \\ b_{s_2ld_2} &= \frac{1}{2} \\ b_{s_3ld_3} &= \frac{2}{3} \\ b(l) &= \frac{2}{3}+\frac{1}{2}+\frac{2}{3}=\frac{11}{6} \end{aligned}$$

∎

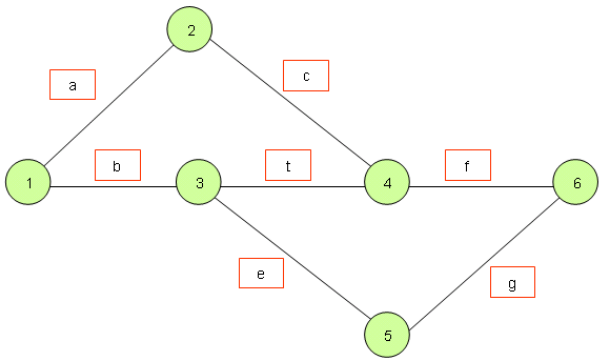


Figure 3.4: Weighted Trap Network

Example 3.2.4 *Consider network of Fig. 3.4 which is called "Trap Network". Details of information about deterministic betweenness is given in Table 3.2. In this table,* $p_{sx}(d)$ *shows the deterministic betweenness of link x for source-destination pair* $s - d$.

The trap topology is well-known in the context of survivable routing. Suppose there is a demand from node 1 for node 6. The min-hop path from node 1 to 6 is the straight line $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$. It appears that this path is the best choice to run the demand, but in survivable routing we need to assign backup paths to each primary route. In trap network there is no link-disjoint backup path for $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$. Therefore it would be beneficial to choose path $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$ ($1 \rightarrow 3 \rightarrow 5 \rightarrow 6$) as the primary route for demands from 1 to 6. Then the link-disjoint backup path will be $1 \rightarrow 3 \rightarrow 5 \rightarrow 6$ ($1 \rightarrow 2 \rightarrow 4 \rightarrow 6$). We will return to the trap topology in chapter 5 and will find its optimal weight assignment to maximize network robustness.

Deterministic link/node betweenness can characterize the topological load of a link/node regardless of the nature of the traffic. We use the available capacity of link l to characterize its load carrying capacity. Now we define our metric to quantify the robustness of a given link.

Definition 3.2.5 *Link Criticality Index (LCI) of link l is the deterministic betweenness of link*

s-d	No. of Paths	$n_{sa}(d)$	$n_{sb}(d)$	$n_{sc}(d)$	$n_{st}(d)$	$n_{se}(d)$	$n_{sf}(d)$	$n_{sg}(d)$
1-2	3	1	2	2	1	1	1	1
1-3	3	2	1	2	1	1	1	1
1-4	3	1	2	1	1	1	1	1
1-5	4	2	2	2	2	2	2	2
1-6	4	2	2	2	2	2	2	2
2-3	3	1	1	2	1	1	1	1
2-4	3	2	2	1	1	1	1	1
2-5	4	2	2	2	2	2	2	2
2-6	4	2	2	2	2	2	2	2
3-4	3	1	1	1	1	1	1	1
3-5	3	1	1	1	1	1	2	2
3-6	3	1	1	1	1	1	2	1
4-5	3	1	1	1	1	2	1	1
4-6	3	1	1	1	1	2	1	2
5-6	3	1	1	1	1	2	2	1
Betweenness		$6\frac{1}{3}$	$6\frac{2}{3}$	$6\frac{2}{3}$	$6\frac{2}{3}$	$6\frac{2}{3}$	$6\frac{2}{3}$	$6\frac{1}{3}$

Table 3.2: Deterministic Betweenness for the Links of Trap Network

l divided by its available capacity (capacity - demand).

More precisely, the link criticality index (LCI) of link $l = (i, j)$ can be written as follows.

$$I(x) = 1 \quad if \quad x > 0 \quad otherwise \quad 0$$
$$LCI(i, j) = \frac{b_{ij}}{c_r(i, j)} \times \frac{1}{I(c_r(i, j) - \gamma_{ij})}$$

where $I(x)$ is the indicator function, $LCI(i, j)$ is the total criticality of link (i, j), $c_r(i, j)$ is the available capacity of link (i, j), b_{ij} is the deterministic betweenness of link (i, j) and γ_{ij} is the present demand on link (i, j). The indicator function is added in the denominator to guarantee that if the demand is more than the available capacity of the link, the demand is not accepted in this link (the link criticality would be effectively infinite in this situation). Definition of the link criticality is clearly showing that the criticality of a link is increasing if more load is carried through this link.

LCI captures the effects that we would like to quantify. One can see that betweenness centrality captures the effect of load. The higher the link betweenness, the more the chance of congestion. On the other hand, the available capacity has an inverse effect on the congestion.

3.2.3 Path Criticality Index (PCI)

The Path Criticality Index (PCI) is defined as the maximum of the LCI of the links along the path.

Definition 3.2.6 *Path Criticality Index (PCI) of a simple path $\pi_{sd} = (i_0 = s, i_1, ..., i_{q-1}, i_q = d)$ is defined as:*

$$PCI(\pi_{sd}) = \max\left(LCI(s, i_1), LCI(i_1, i_2),\ ...,\ LCI(i_{q-1}, d)\right)$$

where π_{sd} is a given path from node s to node d, q is the number of links of the path, and $i_0 = s,\ i_q = d$.

Our definition of *PCI* was first based on the average of the *LCI*s, but our simulations showed that max operator is more effective particularly in heavy traffic. A theory that justifies this choice of *PCI* will come in subsequent chapters.

3.3 Path Criticality Routing Algorithm

The basic idea of our routing algorithm is to accommodate new connection requests along paths that have a low PCI. This requires that we find the link criticality indices. To do this, we need to obtain all possible paths for each source-destination pair. This is not feasible since the number of paths grows rapidly with the number of network nodes and links.

Although the shortest path is not necessarily the path with the lowest PCI, one can expect that the path or paths with lowest PCI are among the k-shortest paths of the network. Hence we use the k-shortest path method proposed by Eppstein [85] with a modification to avoid loops.

A flowchart of our algorithm, *Path Criticality Routing (PCR)*, is shown in Fig. 3.5. PCR algorithm begins with a predefined value of k (default is 1), which may be increased during the course of running the routing algorithm if the desired number of paths to route the traffic cannot not be found. We use thresholds tr_1 (the default value is infinity) and tr_2 (the default value is zero) for PCI. The first threshold defines the lower confidence boundary for the path criticality index. All the paths with path criticality index less than tr_1 are considered eligible to route traffic. On the other hand all the paths with the criticality index larger than tr_2 are considered too risky and may be identified to the re-planning block of AutoNet (via long-loop) for possible capacity enhancement (see Fig. 3.2). If there is no path whose *PCI* is less than tr_1, the paths with criticality index in between the thresholds will carry traffic based on their criticality index as long as they remain within the boundaries. These thresholds in fact define a

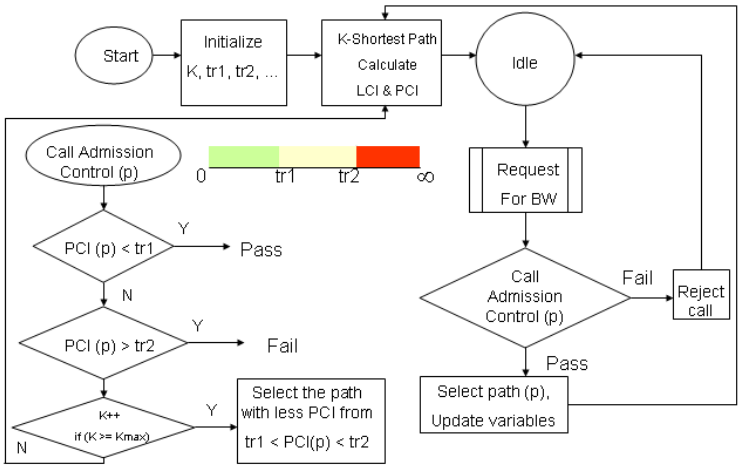


Figure 3.5: PCR Algorithm

simple Call Admission Control (CAC) mechanism that avoids accepting inappropriate traffic into the system.

We note that when a path accepts traffic, the residual capacity of its links will decrease by the carried flow for the duration of the traffic flow. This means that the criticality index of this path must be increased. In other words a constant monitoring of the *PCI* for all the paths is necessary (not real-time necessarily but in reasonable time slots).

In this thesis we are not interested in finding exact methods to determine thresholds tr_1, tr_2. We just mention that different methods can be used to find appropriate values for thresholds in each scenario. One reasonable approach is to use a percentage of link bandwidth as spare part for unpredicted situations and define threshold tr_2 based on it. For example, Cisco considers 25% of a link bandwidth as spare part. The default configuration of link bandwidths are set for maximum utilization of 75%. This limit is programmable and can be changed by costumer. Threshold tr_2 can be considered as the *PCI* of a path when the network utilization is 75% (each link has utilized 75% of its bandwidth). In this case the average of *LCI* is increased to $4 \times \bar{LCI}_{in}$, where

$L\bar{C}I_{in}$ is the initial average of *LCI* (this is the average of *LCI* when the network is not loaded). Therefore, we can choose $tr_2 = 4 \times P\bar{C}I_{in}$, where $P\bar{C}I_{in}$ is the average of *PCI* when network is not loaded. A path can be considered eligible to be assigned to the demand, when its utilization is less than 25%. This in turn defines threshold tr_1 as $tr_1 = \frac{4}{3} \times P\bar{C}I_{in}$.

3.3.1 Time-Complexity of PCR

The most time-consuming part of the algorithm is the k-shortest path calculation. According to [85] the complexity of the proposed k-shortest path algorithm is $O(m + nlogn + k)$ where, m is the number of links and n is the number of nodes. The algorithm is polynomial time and as a result PCR is also a polynomial time algorithm if we set a maximum value for k such as k_{max}. The complexity of the other parts of the algorithm (without k-shortest path) is $O(m^2)$. Therefore the time complexity of PCR will be: $O((m + nlogn + k_{max})m^2)$.

3.4 Evaluation of PCR

We implemented PCR algorithm in C++ and tested it for many network configurations. Here we use four benchmark topologies as well as a realistic topology to compare the effectiveness of PCR with Shortest-Path (SP) and Widest Shortest Path (WSP). We did not have access to the source code (or executable file) of other traffic engineering methods such as MIRA and PBR.

3.4.1 Parking Lot

The parking-lot network topology, shown in Fig. 3.6, is an interesting example. The capacity of all the links are 1. If one unit of bandwidth is requested to be sent from S_0 to D_0 , all the previous routing algorithms (SP, WSP and MIRA) will choose the straight

path and run the flow resulting in the blocking of demands of one unit come from any other source S_i to the destination D_i [25]. A wiser decision is to block the first request from S_0 to D_0 so the network will be able to route the other n requests. In order to do this we need to have appropriate thresholds as discussed in PCR algorithm. We choose $tr_1 = 20$ and $tr_2 = 25$. This choice of thresholds will block demand from S_0 to D_0 since its *PCI* is more than threshold tr_2 (according to table 3.3), but will pass requests from other source-destination pairs as their *PCI* is less than tr_1.

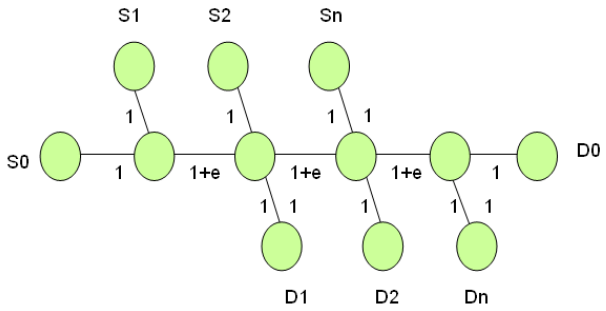


Figure 3.6: Parking-lot Topology n=10, PCR Parameters: $tr_1 = 20$, $tr_2 = 25$, $k_{max} = 3$

.

Path	PCI
$S_0 - D_0$	27.590902
$S_1 - D_1$	11.333333
$S_2 - D_2$	14.090907
$S_3 - D_3$	15.909087
$S_4 - D_4$	17.121209
$S_5 - D_5$	17.727271
$S_6 - D_6$	17.727271
$S_7 - D_7$	17.121209
$S_8 - D_8$	15.909087
$S_9 - D_9$	14.090907
$S_{10} - D_{10}$	11.3333330

Table 3.3: PCI for Different Paths in Parking-Lot

In chapter 5 we will return to the parking-lot topology and will optimize it for maximum robustness.

3.4.2 Concentrator Network

The second test network to study the effectiveness of PCR in determining critical situations is shown in Fig. 3.7, which is called concentrator topology [25]. We consider the following scenario: an online sequence of $n + 1$ requests arrive in sequence for source-destination pairs $S_0 - D$, $S_1 - D$, ... $S_n - D$. The first request has bandwidth requirement n, all others have bandwidth requirement 1. Applying shortest path routing,widest shortest path routing, and minimum interference routing algorithm, one can easily see that all of these methods will choose the shortest path ($S_0 - C - D$) but this will leave only one unit of bandwidth for link (C,D). In other words just one more requests (with 1 unit of bandwidth) can be handled by SP, WSP, and MIRA. All incoming traffic will be blocked during the life-time of this request.

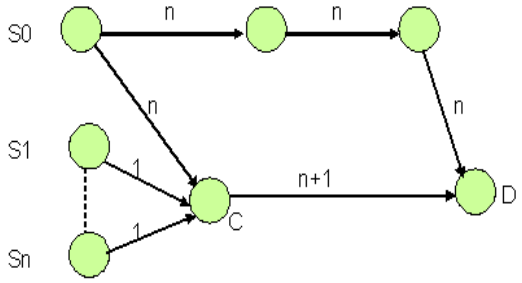


Figure 3.7: Concentrator Topology n=3, PCR Parameters: $k_{max} = 3$

Path	PCI
$S_0 - D$ (3 links)	0.166667
$S_0 - D$ (2 links)	0.520833
$S_1 - D$	0.937500
$S_2 - D$	0.937500
$S_3 - D$	0.937500

Table 3.4: PCI for Different Paths in Concentrator

Table 3.4 reflects the result of our tests on concentrator topology. It is clear that the PCI of path ($S_0 \rightarrow C \rightarrow D$) (or the path with two links from S_0 to D) is much more than the longer path. So, PCR algorithm will choose the longer path (the path with three links) to save the bandwidth of link C-D for other source-destination pairs.

3.4.3 Distributor Topology

Our next example is the distributor network [25] shown in Fig. 3.8. We consider the following scenario: distributor network receives n requests each asking for one unit of bandwidth from S_0 to D. After this, the network gets n requests each one demanding a unit of bandwidth from S_i to D ($i = 1, 2, ..., n$). Applying shortest path routing,widest shortest path routing, and minimum interference routing algorithm will result in choosing shorter paths from S_0 to D. This will block all demands from other sources (S_i, $i = 1, 2, ...n$) to destination D. Table 3.5 reflects the result of our tests

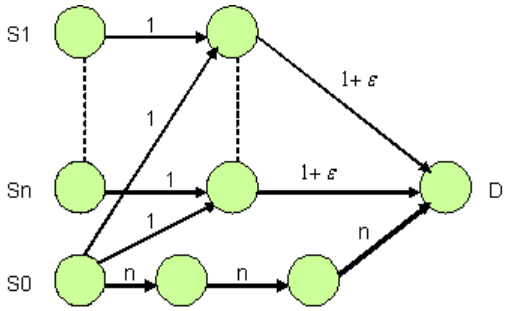


Figure 3.8: Distributor Topology n=3, PCR Parameters: $k_{max} = 3$

Path	PCI
$S_0 - D$ (3 links)	0.083333
$S_0 - D$ (2 links)	0.693182
$S_1 - D$	1.068182
$S_2 - D$	1.068182
$S_3 - D$	1.068182

Table 3.5: PCI for Different Paths in Distributor

on distributor topology. In our algorithm, the path criticality of the longer path is the smallest. Hence this path is dominant and will transport the flow and the blocking problem of the previous algorithms will not be an issue any more.

3.4.4 Rainbow Topology

The fourth network topology of interest is called rainbow. [27] uses this topology to show the difficulty with the profile-based routing algorithm (PBR). The authors in [27] show that the performance of Profile-Based Routing algorithm (PBR) in rainbow

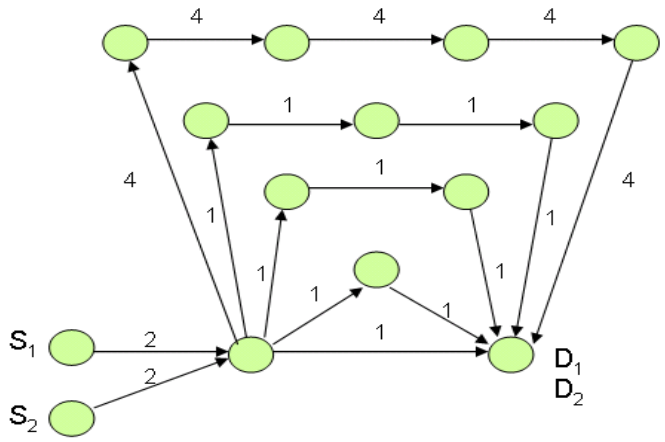


Figure 3.9: Rainbow Topology

network is worse than MIRA, WSP and SP. They have shown that PBR would be blocked after accepting 2 units of bandwidth for $S_1 - D_1$ and 2 units for $S_2 - D_2$.

Path	PCI
$S_1 -> D_1$ (6 links)	0.166667
$S_1 -> D_1$ (5 links)	0.420000
$S_1 -> D_1$ (4 links)	0.425000
$S_1 -> D_1$ (3 links)	0.433333
$S_1 -> D_1$ (2 links)	0.450000

Table 3.6: Rainbow Topology - First Two Demands Arrive

Path	PCI
$S_2 -> D_2$ (6 links)	0.250000
$S_2 -> D_2$ (5 links)	0.420000
$S_2 -> D_2$ (4 links)	0.425000
$S_2 -> D_2$ (3 links)	0.433333
$S_2 -> D_2$ (2 links)	0.450000

Table 3.7: Rainbow Topology - Second Two Demands Arrive

We conducted our experiment in two phases to show the details of the PCR algorithm. Table 3.6 shows the PCI for different paths between source-destination pair $S_1 - D_1$. One can observe that the longest path is the least critical one, hence routing two units of bandwidth. In second step source-destination pair $S_2 - D_2$ will ask for routing two units of bandwidth. At this time the residual capacities are changed and

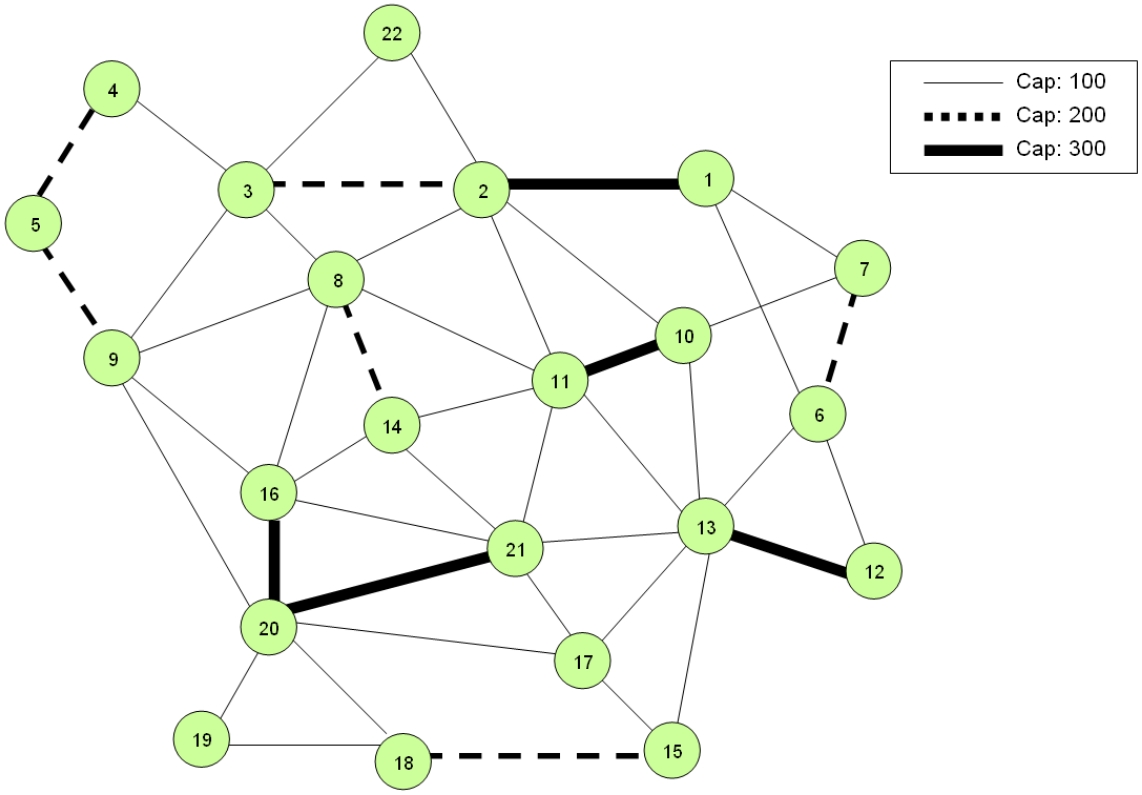


Figure 3.10: Test Network to Evaluate PCR Algorithm

the new PCIs have to be re-calculated as shown in Table 3.7. Again the longest path is still the less critical one and the best candidate for routing the requested flow.

3.4.5 Evaluation of PCR Algorithm on Real Networks

In order to investigate the effectiveness of our PCR algorithm, we ran a set of simulations on the network of Fig. 3.10. We apply PCR to create LSPs (Label Switch Path) assuming that MPLS is used in the network to create the paths.

We considered two scenarios. In the first experiment the requests for LSPs arrive according to a Poisson process and stay for ever (no departures). In our tests the bandwidth requests for paths (LSPs) are taken to be uniformly distributed between 1 to 3 units. In Fig. 3.11 we show the number of rejected requests for this case and

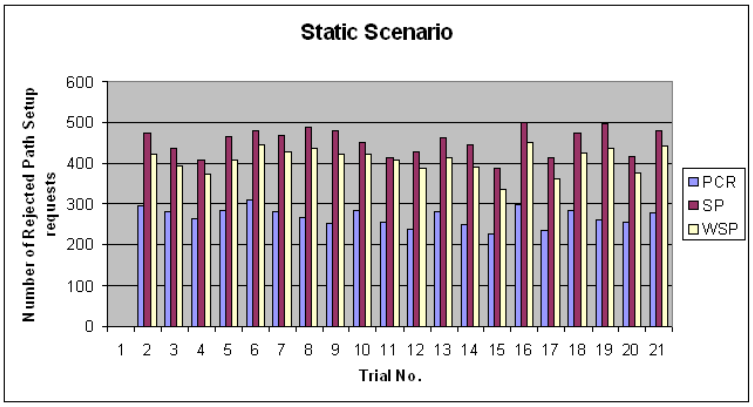


Figure 3.11: Static Scenario- Path Rejection in 2000 Requests, PCR Algorithm

compare the performance to shortest path (SP), and widest shortest path (WSP). The test is performed 20 times, and each test has 2000 path requests. We measured the number of blocked requests.

In a second experiment we examined the behavior of the algorithms in the presence of dynamic traffic. Fig. 3.12 shows the percentage of the path requests rejected in 20 experiments for the following scenario. Path requests arrive between each source-destination point (which is chosen at random) according to a Poisson process with an average rate λ, and the holding times are exponentially distributed with mean μ. We set $\frac{\lambda}{\mu} = 1500$ in our experiments. We generate 7000 requests and measure the rejections or blocking for each one of the algorithms. The results are shown in Fig. 3.12.

In both static and dynamic cases, one can easily see that the PCR has better performance, comparing with WSP and SP algorithms.

3.4.6 AutoNet Software: Integration of PCR with TOTEM

We close this section with a brief description of AutoNet software. The backbone of AutoNet software is TOTEM [12], an open-source toolbox for traffic engineering purposes. TOTEM integrates a series of tools for intra-domain and inter-domain

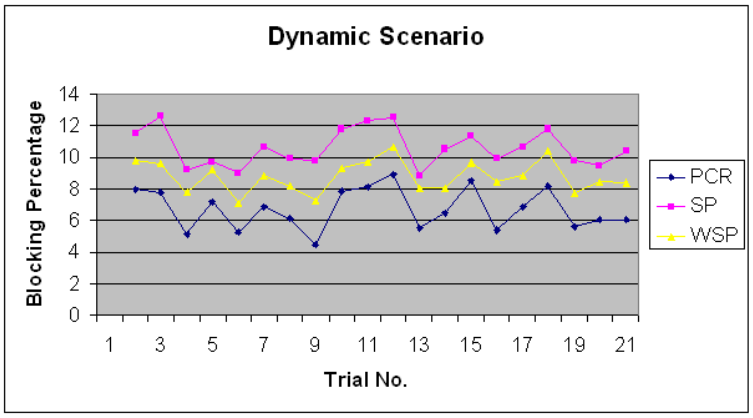


Figure 3.12: Dynamic Scenario- $\frac{\lambda}{\mu}$ = 1500, 7000 Requests

traffic engineering of IP and MPLS networks. It provides an open source software to test different traffic engineering methods. The toolbox is designed to be deployed either as an on-line tool in an operational network, or as an off-line traffic engineering simulator.

We adopted TOTEM as the backbone of AutoNet software. The architecture of TOTEM allows to add various algorithms as a library and TOTEM will see them as an object (object-oriented view). We added PCR to the library of available algorithms in TOTEM. TOTEM provides us with necessary (but not sufficient) graphical tools to visualize different aspects of our developed algorithm.

The part that is missing in TOTEM is the graph-theoretic tools to characterize different graph properties. In addition, integration of traffic matrices and building traffic scenarios is a cumbersome task and needs to be modified.

3.5 Random-Walk Betweenness

The success of the PCR heuristic convinced us that there must be a theoretical basis for its excellent performance, and that this basis must revolve around the notion of betweenness. In this section we attempt to find an explanation for the importance

of betweenness centrality in the analysis of communication networks. Our particular attention is to determine the behavior of a given network (its robustness) in the presence of heavy load (congestion).

Unfortunately the enumeration of paths does not lend itself to tractable analytic results that explain the behavior of PCI. The number of paths between a source-destination pair increases rapidly with the size of networks and there is no mathematical method to calculate the number of paths in general topologies. An upper bound can be found by considering the complete-graph, but even that is not helpful since a simple change in topology can significantly affect the betweenness of nodes or links. However, we have found that the notion of **Random-Walk Betweenness**, introduced by Newman [86], do support the development of a rich set of tractable network optimization algorithms. In order to overcome these problems we used a probabilistic approach to define and analyze the betweenness of a node/link which is based on [86].

Consider a finite-state irreducible Markov Chain with transition probabilities p_{ij} of transitioning from state i at time t to state j at time $t + 1$ (discrete time). The Markov chain can be represented by a state transition diagram with states as nodes in a graph and edges corresponding to allowable transitions, and labels associated with the edges denoting the transition probabilities. The Markov chain can also be viewed as a random walk on the n-node graph with next-step transition probabilities p_{ij}.

We are interested in quantifying the betweenness of a node in the random-walk corresponding to a Markov chain. Consider the set of trajectories that begin at node s and terminate when the walk first arrives at node d, that is, destination node d is an absorbing node. We define the betweenness $b_{sk}(d)$ of node k for the $s - d$ trajectories as the average number of times node k is visited in trajectories from s to d. Note that $b_{dk}(d) = 0$ for k not equal to d since such walks are terminated at step zero.

The path from i to k can be of length 0 to infinity. Let P_d be the matrix of transition probabilities when the random walk is modified so that state d is an absorbing state.

Then ${P_d}^q$ is the matrix of q-step transition probabilities for random walks that terminate at node d. For all $k \neq d$, the probability of entering node k at q^{th} step for a walk that starts at s can be obtained from [sk] element of the matrix ${P_d}^q$. Therefore, the betweenness of a node k for walks from any source s destined for destination d can be obtained by the following matrix equation. Let $B_d = [b_{sk}(d)]$ be the $n \times n$ matrix of betweenness metrics of node k for walks that begin at node s and end at node d. Let $[A]_d$ denote the matrix that results when column d and row d of matrix A are set to zero. Then

$$[B]_d = [b_{sk}(d)]_d = \quad [\sum_{q=0}^{\infty} P_d^q]_d$$

or

$$[B]_d = [b_{sk}(d)]_d = \quad [(I - P_d)^{-1}]_d \tag{3.2}$$

We define the matrix that results from setting the d^{th} diagonal element of the $n \times n$ identity matrix to zero:

$$\Theta_d = [\theta_{sk}(d)] = \begin{cases} 1 & if\ s = k \neq d \\ 0 & otherwise \end{cases}$$

Now from equation 3.2, matrix B_d can be written as:

$$B_d = (I - P_d)^{-1}\Theta_d \tag{3.3}$$

3.6 Packet Networks and Random-Walk Betweenness

We now show that random-walk betweenness is closely related to packet network models. Consider a packet switching network in which packets arrive to packet switches from outside the network according to independent Poisson processes. Each external packet arrival has a specific destination and the packet is forwarded along the network until it reaches said destination. We suppose that packet switches are interconnected by transmission lines that can be modeled as single-server queues.

Furthermore, we also assume that packet switches use a form of datagram routing where a packet at queue i is forwarded to the next-hop queue j with probability p_{ij}.

We calculate the total arrival/departure rate of the traffic to/from each node. The total input rate of node k (internal plus external) is denoted by x_k. After receiving service at the i^{th} node, a customer (random-walk) goes next to node k with probability p_{ik}. To find x_k we need to solve the following set of linear equations (see [87]):

$$x_k = \lambda_k + \sum_{i=1}^{n} x_i p_{ik} \tag{3.4}$$

where λ_k is the arrival rate to node k from outside the network. If we denote $\overrightarrow{x} = [x_1, x_2, ..., x_n]$ and $\overrightarrow{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_n]$, then equation 3.4 becomes:

$$\overrightarrow{x} = \overrightarrow{\lambda} + \overrightarrow{x}P \tag{3.5}$$

Suppose node d is an absorbing node, then we suppose that the arrival rate at node d is zero (since said arrivals do not affect other nodes) and equation 3.5 can be written as:

$$\overrightarrow{x_d} = (\overrightarrow{\lambda_d} + \overrightarrow{x_d}P_d)\Theta_d \tag{3.6}$$

and $\overrightarrow{x_d}$ and $\overrightarrow{\lambda_d}$ are the same as $\overrightarrow{x}$ and $\overrightarrow{\lambda}$ except for the d^{th} element which is 0. Matrix P_d is also the same as P except that its d^{th} row and d^{th} column are zero vectors. Equation 3.6 can be solved for $\overrightarrow{x_d}$.

$$\overrightarrow{x_d} \times (I - P_d \times \Theta_d) = \overrightarrow{\lambda_d} \times \Theta_d$$

Finally:

$$\overrightarrow{x_d} = \overrightarrow{\lambda_d} \times \Theta_d \times (I - P_d \times \Theta_d)^{-1} \tag{3.7}$$

To find the relationship of betweenness B_d and the input arrival rate x_k we notice that $p_{dk}(d) = 0$ which means that $P_d = \Theta_d \times P_d$. Thus:

$$\begin{aligned} P_d \times \Theta_d &= \Theta_d \times P_d \times \Theta_d \\ \Theta_d - P_d \times \Theta_d &= \Theta_d - \Theta_d \times P_d \times \Theta_d \\ (I - P_d) \times \Theta_d &= \Theta_d \times (I - P_d \times \Theta_d) \end{aligned}$$

or

$$\Theta_d \times (I - P_d \times \Theta_d)^{-1} = (I - P_d)^{-1} \times \Theta_d$$

Using equation 3.3 we will have:

$$\Theta_d \times (I - P_d \times \Theta_d)^{-1} = B_d \tag{3.8}$$

We substitute equation 3.8 in 3.7 to find the relationship between the node traffic and node betweenness.

$$\overrightarrow{x_d} = \overrightarrow{\lambda_d} \times B_d \tag{3.9}$$

If we denote the k^{th} element of $\overrightarrow{x_d}$ and $\overrightarrow{\lambda_d}$ by $x_k(d)$ and $\lambda_k(d)$ respectively, we have:

$$x_k(d) = \sum_s \lambda_s(d) b_{sk}(d) \tag{3.10}$$

Equation 3.9 shows that the total arrival rate on node k which is destined for node d is a weighted average of external input rate at each node s destined for node d, where the weight is equal to the betweenness of node k for source-destination pair $s - d$.

It is constructive to establish the relationship of node betweenness and node traffic in a more intuitive way. Consider the traffic generated by packets that arrive at s and are destined for d. Each packet in this flow generates $b_{sk}(d)$ arrivals on average at node k. Let $\lambda_s(d)$ be the number of external packets per second that arrive at node s with destination d. Over a large number of such trials, say N, the average number of times node k is visited will be approximately $N \times b_{sk}(d)$. Suppose that it takes T seconds to have N arrivals at node s, then the average number of visits per second to node k is $\frac{N \times b_{sk}(d)}{T} = \lambda_s(d) \times b_{sk}(d)$, since the average arrival rate at s for d is approximately $\frac{N}{T}$.

We only consider external arrivals with destinations other that the originating node so $\lambda_{dd} = 0$. The total traffic $x_{sk}(d)$ generated by the $s-d$ flow at node k is then $\lambda_s(d) \times b_{sk}(d)$, where s is not equal to d. Recalling that $b_{sd}(d) = 1$, we obtain:

$$x_{sk}(d) = \begin{cases} \lambda_s(d) b_{sk}(d) & if \quad s \neq d \;\&\; d \neq k \\ \lambda_s(d) & if \quad s \neq d \;\&\; k = d \\ 0 & if \quad s = d \end{cases}$$

The total traffic into node k is obtained by summing over all s and d, with s not equal to d

$$\begin{aligned} y_k &= \sum_{s,d} x_{sk}(d) \\ &= \sum_{s \neq k} \lambda_s(k) + \sum_{s \neq d} \sum_{d \neq k} \lambda_s(d) b_{sk}(d) \end{aligned} \tag{3.11}$$

The first sum on the right hand side of equation 3.11 is the total network packet arrival rate destined for k, that is, the total flow absorbed at node k. The second term is the total traffic that flows across queue k, that is, the flow through k that originates at nodes other than d and that are not destined for k. This second term, the transit flow through queue k, accounts for the effect of the network topology, so we let x_k denote this flow:

$$x_k = \sum_{d \neq k} \lambda_k(d) b_{kk}(d) + \sum_{s \neq k} \sum_{d \neq k} \lambda_s(d) b_{sk}(d) \tag{3.12}$$

The first sum in equation 3.12 is the arrivals at k destined for d, including revisits. The second sum is the total transit traffic through k that did not originate locally. x_k can be viewed as a measure of betweenness of queue k that takes the different arrival rates into account.

Suppose that different queues have different total external arrival rates but the fraction of external traffic destined for d does not depend on s, that is,

$$\lambda_s(d) = \lambda_s a_d$$

where

$$a_d \geq 0, \ \sum_d a_d = 1$$

The total traffic through queue k is then

$$\begin{aligned} x_k &= \sum_{d \neq k} \lambda_k a_d b_{kk}(d) + \sum_{s \neq k} \lambda_s [\sum_{d \neq k} a_d b_{sk}(d)] \\ &= \lambda_k [\sum_{d \neq k} a_d b_{kk}(d)] + \sum_{s \neq k} \lambda_s [\sum_{d \neq k} a_d b_{sk}(d)] \end{aligned} \tag{3.13}$$

The terms inside the square brackets in equation 3.14 can be viewed as betweenness measures that have been weighted by the differential preferences for destinations according to a_d. These weighted betweenness measures are in turn scaled according to the arrival rates at different queues.

In the case where arriving packets are equally likely to be destined to any destination (other than the arriving node), we have $a_d = \frac{1}{n-1}$, so

$$\begin{aligned} x_k &= \frac{\lambda_k}{n-1}[\sum_{d\neq k} b_{kk}(d)] + \sum_{s\neq k} \frac{\lambda_s}{n-1}[\sum_{d\neq k} b_{sk}(d)] \\ &= \lambda_k \bar{b}_{kk} + \sum_{s\neq k} \lambda_s \bar{b}_{sk} \end{aligned} \tag{3.14}$$

Finally suppose that the arrival rate at every node is equal, that is, $\lambda_s = \frac{\lambda}{n}$, where λ is the total external packet arrival rate to the network then

$$\begin{aligned} x_k &= \frac{\lambda}{n(n-1)}(\sum_{d\neq k} b_{kk}(d)) + \frac{\lambda}{n(n-1)} \sum_{s\neq k}\sum_{d\neq k} b_{sk}(d) \\ &= \frac{\lambda}{n(n-1)} b_k \end{aligned} \tag{3.15}$$

where we define b_k as the random walk betweenness for node k:

$$b_k = \sum_{s}\sum_{d\neq k} b_{sk}(d)$$

We have derived the following theorem.

Theorem 3.6.1 *Consider a network with n similar nodes, and assume that the average traffic rate on all of the nodes is $\gamma = \frac{\lambda}{n}$ where λ is the total external input traffic rate of the network. Let x_k be the total arrival rate of a node k and b_k be the total betweenness of this node, then:*

$$x_k = \frac{\gamma}{n-1} b_k = \frac{\lambda}{n(n-1)} \times b_k$$

3.6.1 Betweennness and Mean Packet Delay Across the Network (Jackson Networks)

By applying Littles formula to the entire packet network as well as to each of its queues, it can be shown that the average packet delay across the network is given by (see [87]):

$$E[T] = \frac{1}{\lambda} \sum_k x_k E[T_k]$$

where x_k is the total arrival rate to queue k (in packets per second), $E[T_k]$ is the average delay in queue k, and $E[T]$ is the average network delay.

Assume that packets arrive to the network according to a Poisson process, that service times are exponential, and that the service time of a packet at one queue is independent of its service times at other queues, then the network of queues is a Jackson network [87] and the average delay in queue k is given by

$$E[T_k] = \frac{1}{c_k - x_k}$$

where c_k is the total transmission rate of node k (also in packets per second), therefore

$$E[T] = \frac{1}{\lambda} \sum_k \frac{x_k}{c_k - x_k}$$

From Littles formula we also have that the number of packets in the entire network is given by:

$$E[N] = \lambda E[T] = \sum_k \frac{x_k}{c_k - x_k} \tag{3.16}$$

In order to investigate equation 3.16 we consider two extreme cases.

3.6.1.1 Case 1: Light Traffic

When the network is lightly loaded we can assume that $\lambda \to 0$. This means that node traffic x_k is much smaller than the transmission rate of the node ($x_k \ll c_k$). In this case equation 3.16 can be written as:

$$E[N] \to \sum_k \frac{x_k}{c_k}$$

Let $\rho_k = \frac{x_k}{c_k}$ (ρ_k is the *utilization of node k*), and $\bar{\rho} = \frac{\sum_k \rho_k}{n}$ (n is the number of nodes), then we have:

$$E[N] \to \sum_k \rho_k = n\bar{\rho} \tag{3.17}$$

$$E[T] \to \frac{1}{\lambda} \sum_k \rho_k = \frac{n}{\lambda}\bar{\rho} \tag{3.18}$$

3.6.1.2 Case 2: Heavy Traffic

When the network is heavily loaded, equation 3.16 can be estimated as:

$$E[N] \simeq \max_k \frac{x_k}{c_k - x_k} = \max_k \frac{\rho_k}{1 - \rho_k}$$

$$E[N] \to \frac{1}{\frac{1}{\max_k \rho_k} - 1} \tag{3.19}$$

In other words there is a bottleneck node in heavy traffic that dominates $E[N]$. This node is the one with maximum utilization ρ_k. Note that we assume $c_k \geq x_k$ for all the network nodes.

Equations 3.17 and 3.19 provide some directions to the network design problem in the sense that they show the network condition under light and heavy traffic. In lightly loaded networks, in order to minimize the average number of packets in the network we need to minimize the average node utilization, whereas in heavily loaded networks the maximum node utilization should be minimized.

Note that in all of these results the node betweenness plays an important role because the node traffic x_k is a function of node betweenness. To see this in a more clear way consider the special case, where the average external traffic of all the nodes are equal to $\gamma = \frac{\lambda}{n}$ (λ is the total external traffic of the network). Then we can use theorem 3.6.1 to write equations 3.17 and 3.19 based on the node betweenness. For lightly loaded scenario, we have:

$$E[N] \to \sum_k \rho_k = \frac{\gamma}{n-1} \sum_k \frac{b_k}{c_k}$$

One can see that $\frac{b_k}{c_k}$ is very similar to the definition of LCI (see definition 3.2.5). We call $\frac{b_k}{c_k}$ the Node Criticality Index (*NCI*) of k. We see that in case of light traffic, in order to minimize network delay, one needs to minimize the average of *NCI* of the network.

In case of heavy traffic we need to minimize the maximum ρ_k to get the least network delay for packets in the network. But:

$$\rho_k = \frac{x_k}{c_k} = \frac{\gamma}{n-1}\frac{b_k}{c_k}$$
$$\max_k \rho_k = \frac{\gamma}{n-1}\max_k(\frac{b_k}{c_k})$$

This means that we need to minimize the maximum *NCI* of a network to have the best delay performance. We summarize these results in the following theorem.

Theorem 3.6.2 *Consider a network in which the packets arrive according to a Poisson process, service times are exponential, and the service time of a packet at one queue is independent of its service times at other queues. In order to minimize the average network delay when the network is heavily loaded, the maximum of NCI of the network should be minimized. Further, when the network is lightly loaded, the average NCI should be minimized to achieve the best average delay performance.*

In chapter 4, which is entirely dedicated to the concept of criticality, we extend the idea of NCI to a general weighted graph and derive useful interpretations for criticality.

Theorem 3.6.2 provides useful guidelines for designing robust networks. In case of light load a network should solve an optimization problem which tries to minimize the average node criticality of the network, while in scenarios with heavy traffic one should consider networks that minimizes the maximum node criticality of the network. We will integrate these two optimization problems into one general convex optimization problem later in this thesis.

Chapter 4

Network Criticality using Betweenness Metrics

This chapter develops the main theoretical foundation for the metric to quantify the **survival value** of a network. To do this, we generalize the definition of LCI (link criticality index) and introduce link/node criticality for weighted graphs. We show that the criticality of any given node/link in a graph has a common global measure and a specific local measure. The global quantity is what we call **network criticality** and will be used to quantify the survival value of a network. This metric is an indicator of the robustness of a network.

4.1 Network Model

We model a network with an undirected weighted graph $G = (N, E, W)$ where N is the set of nodes, E is the set of graph links, and W is the weight matrix of the graph (a weight is associated with each link of the graph). Throughout this thesis we assume that ***G is a connected graph***.

The random-walk betweenness is determined by the transition probabilities of the

walk, that is the probability of proceeding to a given neighbor when the walk is at a specified node. We will suppose that the transition probabilities are defined by the weights as follows.

$$p_{ij} = \begin{cases} \frac{w_{ij}}{\sum_{k \in A(i)} w_{ik}} & if \quad j \in A(i) \\ 0 & otherwise \end{cases} \tag{4.1}$$

where $A(i)$ is the set of adjacent nodes of i and w_{ik} is the non-negative weight of link (i, k) if any. Equation 4.1 defines a Markov chain on graph $G = (N, E, W)$. In this thesis we are interested in random-walks with specified source and destinations, where the destination of the walk is an absorbing state of the Markov chain, that is node d is an absorbing node, and any random-walk coming to this state, will be absorbed. Let $p_{sk}(d)$ be the probability of proceeding to a neighbor k when the walk is at node s and its final destination is node d. We can write $p_{sk}(d)$ based on equation 4.1 as follows:

$$p_{sk}(d) = p_{sk}(1 - \delta_{sd}) \tag{4.2}$$

where δ_{kd} denotes Kronecker delta function. Note that $p_{dk}(d) = 0$.

Observation 4.1.1 *Equation 4.1 shows that if the weight increases, the desirability or goodness of that link (probability of being chosen) also increases. In the context of QoS this means that QoS parameters for which larger value denotes goodness should be positively related to the weight. We call these parameters "beneficial QoS parameters". In contrast, the QoS parameters for which increasing value denotes decreasing goodness, are called "detrimental QoS parameters".*

Mathematically speaking, the derivative of the link weight with respect to a beneficial QoS parameter is non-negative, while the derivative of the link weight with respect to a detrimental QoS parameter is non-positive.

Example The following metrics are examples of beneficial and detrimental QoS parameters.

- Capacity is a beneficial QoS parameter.
- Available bandwidth is a beneficial QoS parameter.
- Used bandwidth is a detrimental QoS parameter.
- Packet loss is a detrimental QoS parameter.

Observation 4.1.1 suggests that SLA parameters can be related to the link weights. In this thesis we are interested in the study of the weight and its effect on robustness. We assume that SLA parameters are already mapped to the weights with an appropriate method. Some of these methods are discussed in [88]. This permits us to abstract different business policies and/or SLA's as parts of the weight definition. This is indeed an important feature of an autonomic system that differentiates it from an adaptive system, which needs to be re-designed when business policies change [82].

One way of mapping QoS parameters to the link weights is as follows.

$$\begin{aligned} w_{ij} &= w_{ij}^{qos_1} \times w_{ij}^{qos_2} \times ... w_{ij}^{qos_k} \\ &= \frac{w_{ij}^{(1)}}{w_{ij}^{(a_1)}} \times \frac{w_{ij}^{(2)}}{w_{ij}^{(a_2)}} \times ... \frac{w_{ij}^{(k)}}{w_{ij}^{(a_k)}} \end{aligned} \tag{4.3}$$

where w_{ij}^{q} is a beneficial QoS parameter and $w_{ij}^{a_q}$ is a detrimental QoS parameter.

4.2 Betweenness as a Function of Weights

In chapter 3 we established the relationship between the probability transition matrix P_d and betweenness matrix B_d in equation 3.3. One can see that the removal of column and row d from the betweenness and probability matrices does not affect the other equations. We use $P(i|j)$ to show the truncated $(n-1) \times (n-1)$ matrix that results from removing i^{th} row and j^{th} column of matrix P. Equation 3.3 can be written as:

$$B_d(d|d) = (I - P_d(d|d))^{-1} \tag{4.4}$$

Let W be the weight matrix and let D be the diagonal matrix of weighted graph degrees:

$$\begin{aligned} D &= diag(W_1, W_2, ..., W_n) \\ W_i &= \sum_{k=1}^{n} w_{ik} \end{aligned}$$

Note also that $P_d(d|d) = D^{-1}(d|d)W(d|d)$. Therefore:

$$\begin{aligned} I - P_d(d|d) &= I - D^{-1}(d|d) \times W(d|d) \\ &= D^{-1}(d|d) \times (D(d|d) - W(d|d)) \end{aligned}$$

But $L = D - W$ is the **Laplacian matrix** of the graph [89, 90, 91, 92] (A short review of the graph Laplacian is given in appendix A). Therefore

$$\Rightarrow I - P_d(d|d) = D^{-1}(d|d) \times L(d|d) \tag{4.5}$$

Substitution of equation 4.5 in 4.4 results in:

$$B_d(d|d) = L^{-1}(d|d) \times D(d|d) \tag{4.6}$$

Note that the graph $G(N, E, W)$ is assumed to be connected which means that the rank of graph Laplacian L is $(n - 1)$. As a result, the inverse of the reduced Laplacian $L(d|d)$ exists and equation 4.6 has a unique solution.

We will now develop equation 4.6 to obtain an expression in terms of the original Laplacian of the graph. We use lowercase Roman letters to denote column vectors and lowercase Greek letters to denote row vectors. We use subscripts to specify the dimension of a vector. For example z_{n-1} is a $(n - 1) \times 1$ column vector and ζ_{n-1} is a $1 \times (n - 1)$ row vector.

Without loss of generality, we relabel the nodes so that the removed node becomes the last node of the graph (node n).

Now, in order to write $L^{-1}(n|n)$ in terms of L, we use the Moore-Penrose generalized inverse matrix of L [93]. The Moore-Penrose inverse of $L(n|n)$ and the matrix $L^{-1}(n|n)$

are equal since $L(n|n)$ is an $(n-1)\times(n-1)$ matrix with rank $n-1$. In other words, $L(n|n)$ is full-rank and its inverse is the same as its Moore-Penrose inverse. To obtain L from $L(n|n)$, we first add a column to $L(n|n)$ to get: $Q = [L(n|n)\ z_{n-1}]$.

One important property of the Laplacian matrix of a graph is that the sum of any row of the Laplacian matrix is zero. Therefore, the column-vector z_{n-1} has to be chosen in a way to make the sum of every row of the matrix Q equal to zero. We use the following recursive formula from [93] to obtain the Moore-Penrose inverse of a matrix when a column is added to the original matrix. Let A be a $p\times q$ matrix and b_p be a $p\times 1$ column vector.

$$\begin{pmatrix} A & b_p \end{pmatrix}^+ = \begin{pmatrix} A^+(I - b_p\zeta_p) \\ \zeta_p \end{pmatrix} \tag{4.7}$$

where ζ_p is a $1\times p$ row vector such that

$$\zeta_p = \begin{cases} b_p - AA^+b_p & if\ \ b_p \neq AA^+b_p \\ \frac{b_p^*(AA^*)^+}{1+b_p^*(AA^*)^+b_p} & if\ \ b_p = AA^+b_p \end{cases} \tag{4.8}$$

where $*$ denotes *conjugate transpose* of a matrix. To satisfy the requirement of Laplacian matrix we need to have

$$[L(n|n)\ z_{n-1}]\overrightarrow{1}_n = 0 \tag{4.9}$$

where $\overrightarrow{1}_n$ is a $n\times 1$ vector of all ones: $\overrightarrow{1}_n = [1\ 1\ 1\ ...\ 1]^t$.
From 4.9 one can see that:

$$\begin{aligned} L(n|n)\overrightarrow{1}_{n-1} + z_{n-1} &= 0 \\ z_{n-1} &= -L(n|n)\overrightarrow{1}_{n-1} \end{aligned} \tag{4.10}$$

Now from 4.7 by replacing $A = L(n|n)$, $p = n-1$, $b_{n-1} = z_{n-1}$ and using 4.10, we have:

$$\begin{aligned} Q^+ &= \begin{pmatrix} L(n|n) & z_{n-1} \end{pmatrix}^+ \\ &= \begin{pmatrix} L^+(n|n) - L^+(n|n) z_{n-1} \zeta_{n-1} \\ \zeta_{n-1} \end{pmatrix} \\ &= \begin{pmatrix} L^+(n|n) + L^+(n|n) L(n|n) \overrightarrow{1}_{n-1} \zeta_{n-1} \\ \zeta_{n-1} \end{pmatrix} \\ &= \begin{pmatrix} L(n|n)^+ + \overrightarrow{1}_{n-1} \zeta_{n-1} \\ \zeta_{n-1} \end{pmatrix} \\ &= \begin{pmatrix} L(n|n)^+ \\ 0 \end{pmatrix} + \overrightarrow{1}_{n} \zeta_{n-1} \end{aligned}$$

This expression for Q^+ can be expanded as:

$$Q^+ = \begin{pmatrix} L(n|n) & z_{n-1} \end{pmatrix}^+ = \begin{pmatrix} L^+(n|n) \\ 0 \end{pmatrix} + \begin{pmatrix} \zeta_{n-1} \\ \zeta_{n-1} \\ . \\ . \\ . \\ \zeta_{n-1} \end{pmatrix} \tag{4.11}$$

Equation 4.11 asserts that the [sk] and [nk] element of Q^+ are:

$$\begin{aligned} q^+_{sk} &= (L^+(n|n))_{sk} + (\zeta_{n-1})_k \\ q^+_{nk} &= 0 + (\zeta_{n-1})_k \end{aligned}$$

Subtracting these two equations gives the [sk] element of $L^+(n|n)$:

$$\Rightarrow (L^+(n|n))_{sk} = q^+_{sk} - q^+_{nk} \tag{4.12}$$

With the same approach , we can add the n^{th} row to Q to obtain the $n \times n$ Laplacian matrix L: $\mathrm{L} = \begin{bmatrix} Q \\ d \end{bmatrix}$

With a similar derivation and using equation 4.7 one can obtain:

$$\Rightarrow q_{sk}^{+} = l_{sk}^{+} - l_{sn}^{+} \tag{4.13}$$

Combining equations 4.12, 4.13 we can find our desired result:

$$(L^{+}(n|n))_{sk} = l_{sk}^{+} - l_{sn}^{+} - l_{nk}^{+} + l_{nn}^{+} \tag{4.14}$$

∎

Now, according to equation 4.6, we can obtain the betweenness of the node k for source-destination pair (s, d):

$$\begin{aligned} B_d(d|d) &= L^{-1}(d|d) \times D(d|d) \\ (B_d(d|d))_{sk} &= (l_{sk}^{+} - l_{sd}^{+} - l_{dk}^{+} + l_{dd}^{+}) \times W_k \\ \frac{b_{sk}(d)}{W_k} &= l_{sk}^{+} - l_{sd}^{+} - l_{dk}^{+} + l_{dd}^{+} \end{aligned} \tag{4.15}$$

Lemma 4.2.1 *For a network with transition probabilities given by equation 4.1, there is a simple relationship between $b_{sk}(d)$ and $b_{dk}(s)$: $\frac{b_{sk}(d)+b_{dk}(s)}{W_k} = l_{dd}^{+} + l_{ss}^{+} - 2l_{sd}^{+}$*

Proof According to equation 4.15 we have:

$$\begin{aligned} \frac{b_{sk}(d)}{W_k} &= l_{sk}^{+} - l_{sd}^{+} - l_{dk}^{+} + l_{dd}^{+} \\ \frac{b_{dk}(s)}{W_k} &= l_{dk}^{+} - l_{ds}^{+} - l_{sk}^{+} + l_{ss}^{+} \\ \frac{b_{sk}(d) + b_{dk}(s)}{W_k} &= l_{ss}^{+} + l_{dd}^{+} - l_{ds}^{+} - l_{sd}^{+} \\ \frac{b_{sk}(d) + b_{dk}(s)}{W_k} &= l_{ss}^{+} + l_{dd}^{+} - 2l_{sd}^{+} \end{aligned}$$

The last equation is a direct result of the symmetry of the Laplacian matrix and its generalized inverse. ∎

Lemma 4.2.1 shows that $\frac{b_{sk}(d)+b_{dk}(s)}{W_k}$ is independent of node k.

Definition 4.2.2 *We define $\tau_{sd} = l_{ss}^{+} + l_{dd}^{+} - 2l_{sd}^{+}$ as the criticality of source-destination pair $s - d$ or $d - s$.*

Lemma 4.2.1 gives an interpretation of τ_{sd} based on the betweenness of any node k for pair $s-d$ or $d-s$. The criticality between node s and d is the betweenness of any node k per unit of node weight for pair $s-d$ or $d-s$. Note that $\tau_{sd} = \tau_{ds}$ since L^+ is a symmetric matrix.

Finally we obtain the total betweenness of node k. We notice that $\sum_s \sum_d b_{sk}(d) = \sum_s \sum_d b_{dk}(s)$. This can easily be verified by changing subscript s with d. Hence:

$$\begin{aligned} \frac{b_k}{W_k} &= \frac{1}{W_k} \sum_s \sum_d b_{sk}(d) = \frac{1}{W_k} \sum_s \sum_d \frac{b_{sk}(d) + b_{dk}(s)}{2} \\ &= \sum_s \sum_d \frac{\tau_{sd}}{2} \\ &= \frac{1}{2} \sum_s \sum_d (l^+_{ss} + l^+_{dd} - 2l^+_{ds}) \end{aligned} \tag{4.16}$$

4.3 Network Criticality

Now, we are ready to extend our metric LCI to a more general quantity.

Definition 4.3.1 *The criticality of a node k is defined as the ratio of the node betweenness to its total weight:*

$$\eta_k = \frac{b_k}{W_k} = \frac{1}{2} \sum_s \sum_d \tau_{sd} = \frac{1}{2}\tau \tag{4.17}$$

where

$$\tau_{sd} = l^+_{ss} + l^+_{dd} - 2l^+_{sd} \tag{4.18}$$

$$\tau = \sum_s \sum_d \tau_{sd} \tag{4.19}$$

We give another representation of τ_{sd} which will be used in subsequent chapters. Let u_i be the i^{th} unit vector, and define $u_{ij} = u_i - u_j$. Since $u_i = [0\ 0\ ...\ 1\ ...\ 0]^t$, where the 1 is in i^{th} position and $u_{ij} = [0\ ...\ 1\ ...\ -1\ ...\ 0]^t$, where 1 and -1 are in i^{th} and j^{th} position.

Now equation 4.18 can be written as:

$$\tau_{sd} = u_{sd}^t L^+ u_{sd} \tag{4.20}$$

Definition 4.3.2 *We define criticality of an edge or link l=(i,j) as the betweenness of the node over its weight:* $\eta_{ij} = \eta_{ij} = \frac{b_{ij}}{w_{ij}}$

Lemma 4.3.3 *The criticality of a link l=(i,j) is equal to* τ *:*

$$\eta_{ij} = \frac{b_{ij}}{w_{ij}} = \tau \tag{4.21}$$

Proof The betweenness of link $l = (i, j)$ can be extracted from the betweenness of its end nodes i,j. If b_i is the betweenness of node i, then the share of link (i, j) is on average $b_i \times p_{ij}$ in $i \to j$ direction, where p_{ij} is the probability of transitioning from i to j in next step. Similarly, if the betweenness of node j is b_j, then the share of link (i, j) in $j \to i$ direction will be $b_j \times p_{ji}$. Therefore total betweenness of link (i, j) is:

$$\begin{aligned}
b_{ij} &= b_i \frac{w_{ij}}{\sum_k w_{ik}} + b_j \frac{w_{ji}}{\sum_k w_{jk}} \\
&= \frac{1}{2}\tau \sum_k w_{ik} \frac{w_{ij}}{\sum_r w_{ir}} + \frac{1}{2}\tau \sum_k w_{jk} \frac{w_{ji}}{\sum_r w_{jr}} \\
&= \frac{1}{2}\tau(w_{ij} + w_{ji}) = \tau w_{ij} \\
b_{ij} &= \tau w_{ij} \quad \forall (i, j) \in E \\
\frac{b_{ij}}{w_{ij}} &= \tau \quad \forall (i, j) \in E
\end{aligned} \tag{4.22}$$

Where w_{ij} and η_{ij} show the weight and the criticality of link $l = (i, j)$ respectively. ∎

Observation 4.3.4 *One can see from 4.17 and 4.21 that the criticality of a node k or a link* (i, j) *is independent of its position and is proportional to* τ*.*

Observation 4.3.4 is a significant result showing that the betweenness of a node (link) can be written as the product of two graph values ($b_k = W_k \frac{\tau}{2}$ for a node k or $b_{ij} = w_{ij}\tau$ for a link (i, j)) one of which values is a local metric i.e. the weighted degree

of a node (or weight of a link), and the other of which values is a network-wide metric τ which is only a function of the graph weight matrix. This global metric is our main tool to investigate the robustness of different networks. The smaller the value of τ, the better the robustness of a network. Indeed τ is the *survival value* that we postulated in our discussion in Chapter 3 because it can be used to quantify the resistance of a network to the unwanted changes in network topology or traffic demands. The smaller the value of τ, the less the sensitivity to the changes in topology and traffic. Note that increasing the betweenness of a node (link) decreases the robustness, whereas increasing the beneficial weight of a node (link) increases the network robustness as well. We will shed more light on this matter later in this chapter by providing different interpretations of τ and through extensive examples.

Definition 4.3.5 *τ is defined as the* **network criticality** *for $G(N, E, W)$. The average (or normalized) network criticality is also defined as: $\hat{\tau} = \frac{\tau}{n(n-1)}$.*

The average (normalized) network criticality quantifies the average criticality of any two pairs of nodes in a network. Therefore $\hat{\tau}$ is useful for robustness comparison between network topologies. A network topology is more robust if its $\hat{\tau}$ is less than another network topology.

Network criticality can also be used to design methods to engineer the evolution of network flows or network topology. Chapter 5 of the thesis will focus on the design methodologies to manage the evolution of network topologies or network algorithms.

4.4 Some Facts about Network Criticality

In this section we establish some lemmas which will be central to our other derivations.

Lemma 4.4.1 *Network Criticality τ is equal to $2nTr(L^+)$. Equivalently, average network criticality $\hat{\tau}$ is $\frac{2}{n-1}Tr(L^+)$.*

Proof Since $\tau_{sd} = l^+_{ss} + l^+_{dd} - 2l^+_{sd}$, we have

$$\begin{aligned}
\tau &= \sum_{s,d} \tau_{sd} \\
&= \sum_{d}\sum_{s} l^+_{ss} + \sum_{s}\sum_{d} l^+_{dd} - 2\sum_{s}\sum_{d} l^+_{sd} \\
&= n\sum_{s} l^+_{ss} + n\sum_{d} l^+_{dd} - 2 \times 0 \\
&= 2n\sum_{i} l^+_{ii} \\
\tau &= 2nTr(L^+) \\
\hat{\tau} &= \frac{1}{n(n-1)} 2nTr(L^+) \\
\hat{\tau} &= \frac{2}{n-1} Tr(L^+)
\end{aligned}$$

This completes the proof of lemma 4.4.1. ∎

Lemma 4.4.2 *$\hat{\tau}$ can be written as: $\hat{\tau} = \frac{2}{n-1}\sum_{i=2}^{n} \frac{1}{\lambda_i}$, where $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are eigenvalues of graph Laplacian L.*

Proof We know from linear algebra that the trace of a square matrix is equal to the sum of its eigenvalues. On the other hand, the non-zero eigenvalues of L^+ are reciprocals of the non-zero eigenvalues of L. Lemma 4.4.2 is then a direct result of Lemma 4.4.1. ∎

Lemma 4.4.2 will establish a connection between network criticality and the spectrum of graph Laplacian.

Lemma 4.4.3 *For any weight matrix W of links of a graph: $Vec(W)^t \nabla\tau + \tau = 0$, where $Vec(W)$ is a vector obtained by concatenating all the rows of matrix W to get a vector of w_{ij}'s.*

Proof Suppose we scale all the link weights in a graph with factor t, then by lemma 4.3.3

$$\begin{aligned}
\tau(Vec(W)) &= \frac{b_{ij}}{w_{ij}} \\
\tau(tVec(W)) &= \frac{b_{ij}}{tw_{ij}}
\end{aligned}$$

Note that the transition probabilities $p_{sk}(d)$ are invariant to the scaling of the weights based on their definition in equation 4.1. Therefore matrix P_d is also invariant to the scaling of the weights. As a direct result matrix B_d (and each one of its elements $b_{sk}(d)$) is invariant to the scaling of the weights based on equation 3.3. This verifies that any node betweenness $b_k = \sum_{s,d} b_{sk}(d)$ is invariant to the scaling and finally equation 4.22 asserts that any link betweenness b_{ij} is invariant to the scaling of the link weights. Therefore

$$\tau(tVec(W)) = \frac{1}{t}\tau(Vec(W)) \tag{4.23}$$

By taking the derivative of τ with respect to t, we have

$$Vec(W)^t \nabla\tau = \frac{-1}{t^2}\tau(W) \tag{4.24}$$

It is enough to consider equation 4.24 at $t = 1$ to get $Vec(W)^t\nabla\tau + \tau = 0$. ∎

4.5 Some Interpretations of Network Criticality

In this section we try to shed more light on the concept of network criticality by providing some interpretations of it.

4.5.1 Network Criticality and Average Path Cost

We assume certain cost to travel along a path and study the effect of network criticality τ on average cost incurred by a message during its walk from source s to destination d.

We consider the following scenario. For each link $l = (i, j)$ there is a cost $z_l = z(i, j)$ (Fig. 4.1). Note that this cost is different from the weight of the link. After a random-walk starts from source node s, at each step it traverses one link, incurs a cost, and continues until it is absorbed by destination d. We wish to calculate the average cost of

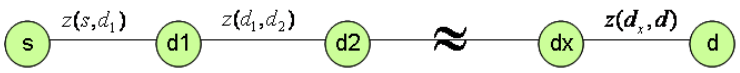


Figure 4.1: Random-walk Movement

this journey. Theoretically, the number of hops of a path that is taken by random-walk can be infinite.

Using the properties of Markov Chains one can calculate the average cost $\varphi(s,d)$ incurred by paths between source s and destination d:

$$\varphi(s,d) = E\{\sum_{k=0}^{\infty} z(d_k, d_{k+1})\} \quad where \;\; s = d_0 \tag{4.25}$$

We expand equation 4.25 using elementary probability.

$$\begin{aligned}
\varphi(s,d) &= \sum_{d_1,d_2,\ldots} p(d_1,d_2,\ldots|d_0 = s)(\sum_{k=0}^{\infty} z(d_k,d_{k+1})) \\
&= \sum_{d_1} p(d_1|d_0)\{\sum_{d_2,d_3,\ldots} p(d_2,\ldots|d_1)(z(s,d_1) + \sum_{k=1}^{\infty} z(d_k,d_{k+1}))\} \\
&= \sum_{d_1} p_{sd_1}\{z(s,d_1) + \sum_{d_2,d_3,\ldots} p(d_2,\ldots|d_1)\sum_{k=1}^{\infty} z(d_k,d_{k+1})\} \\
&= \sum_{d_1} p_{sd_1} z(s,d_1) + \sum_{d1} p_{sd_1}\varphi(d_1,d)
\end{aligned}$$

in brief:

$$\varphi(s,d) = \sum_j p_{sj} z(s,j) + \sum_j p_{sj}\varphi(j,d) \tag{4.26}$$

Equation 4.26 has a recursive form, which we will express in matrix form. We relabel the nodes so that node d is the last node. We also define $f_s = \sum_j p_{sj} z(s,j)$. Equation 4.26 can be written as:

$$\begin{aligned}
\overrightarrow{\phi}_d(d|d) &= \overrightarrow{f}(d|d) + P_d(d|d)\overrightarrow{\phi}_d(d|d) \\
\overrightarrow{\phi}_d(d|d) &= (I - P_d(d|d))^{-1}\overrightarrow{f}(d|d)
\end{aligned} \tag{4.27}$$

where $\overrightarrow{\phi}_d = [\phi(s_1,d), \phi(s_2,d), ..., \phi(s_n,d)]$, $\overrightarrow{f} = [f_{s_1}, f_{s_2}, ..., f_{s_n}]$. Now we can use equation 4.4 to write the cost as a function of betweenness. In order to simplify the notation, we substitute $p_{sk}(d)$ with p_{sk} in the following equations. This is safe because node d is an absorbing node and it does not have any effect on the transition probability of other node pairs. In the following equations we work with reduced matrices where the effect of absorbing state is already considered by removing the row and column corresponding to node d.

$$\begin{aligned} \overrightarrow{\phi}_d(d|d) &= B_d(d|d)\overrightarrow{f}(d|d) \\ or \quad \varphi(s,d) &= \sum_{k=1}^{n} b_{sk}(d) \times f_k \\ &= \sum_{k=1}^{n} b_{sk}(d) \sum_j p_{kj} z(k,j) \end{aligned} \tag{4.28}$$

Equation 4.28 has a nice interpretation. In order to obtain average cost from any node s to any other node d, one needs to find the average cost from a node k to any other node in the network first (i.e. $f_k = \sum_j p_{kj} z(k,j)$). Then the average cost from node s to node d is the product of this cost for node k with the betweenness of node k for source-destination pair (s,d) summed over all nodes. Note that the average cost of any node k to other nodes (f_k) depends on the Markov chain probabilities, while the term $\sum_k b_{sk}(d) f_k$ is in fact a kind of topological average.

Now we are ready to calculate the average cost over all node-pairs.

$$\begin{aligned} \bar{\varphi} &= \frac{1}{n(n-1)} \sum_{s,d} \varphi(s,d) \\ &= \frac{1}{n(n-1)} \sum_k (\sum_{s,d} b_{sk}(d) \sum_j p_{kj} z(k,j)) \\ &= \frac{1}{n(n-1)} \sum_k (\sum_j p_{kj} z(k,j) \sum_{s,d} b_{sk}(d)) \\ &= \frac{1}{n(n-1)} \sum_k (\sum_j p_{kj} z(k,j) b_k) \end{aligned}$$

One can use relation 4.17 to find the relationship between average cost and criticality.

$$\begin{aligned}
\bar{\varphi} &= \frac{1}{n(n-1)} \sum_k (\sum_j p_{kj} z(k,j) \frac{1}{2} \tau W_k) \\
&= \frac{1}{2n(n-1)} \tau \sum_k (\sum_j \frac{w_{kj}}{W_k} z(k,j) W_k) \\
&= \frac{1}{2n(n-1)} \tau \sum_k (\sum_j w_{kj} z(k,j)) \qquad (4.29) \\
&= \frac{1}{2} \hat{\tau} \sum_k (\sum_j w_{kj} z(k,j)) \qquad (4.30)
\end{aligned}$$

Observation 4.5.1 *Equation 4.29 shows that the average network cost is the product of average network criticality and total weighted graph cost* ($\sum_k(\sum_j w_{kj} z(k,j))$). *If* $\sum_k(\sum_j w_{kj} z(k,j))$ *is fixed at constant value C (maximum budget) then* the average network cost is proportional to the criticality of the network. *If we interpret a link weight* w_{kj} *as the price we should pay for a unit of cost for link* (k,j)*, then C can be interpreted as a total budget of the network.*

This interpretation of network criticality is important because in many practical situations we aim to minimize the average cost of a network. For example most of the traffic engineering algorithms try to minimize a kind of cost in the system. Another example is network planning (or re-planning). In network design we have an optimization criteria where a cost metric is minimized. We will see some examples of network planning using τ in chapter 5. Motivated by this observation, in chapter 5 we will investigate the problem of minimizing network criticality by defining some appropriate constraints.

4.5.2 Network Criticality and Average Hop Length

Network Criticality is also related to the average hop length of a walk. The following important result relates the average length of random-walk to the network criticality.

Lemma 4.5.2 *Let T be the average length (number of hops) of a random-walk over all source-destination pairs, and $\bar{W}$ be the average weight of all nodes. Then:*

$$T = \frac{n}{2}\bar{W}\hat{\tau}$$

Proof An special case of cost $z(i, j)$, as defined in section 4.5.1, is the case of equal cost for all the links. While it is possible to derive lemma 4.5.2 by considering $z(i, j) = 1$ for all possible links in equation 4.30, we take an alternative approach, because some parts of the proof will be used later. The average hop count of a random-walk that starts at node s before it reaches its destination node d is equal to the average number of times a node k is visited summed over all possible k's. In terms of betweenness this is:

$$T_{sd} = \sum_{k} b_{sk}(d)$$

Substituting $b_{sk}(d)$ from equation 4.15 results in:

$$T_{sd} = \sum_{k=1}^{n} (l^{+}_{sk} - l^{+}_{sd} - l^{+}_{dk} + l^{+}_{dd})W_k$$

Now, the average time in system considering all possible source-destination pairs would be:

$$\begin{aligned}
T &= \frac{1}{n(n-1)} \sum_{s,d} T_{sd} \\
&= \frac{1}{n(n-1)} \sum_{s,d} \sum_{k=1}^{n} (l^{+}_{sk} - l^{+}_{sd} - l^{+}_{dk} + l^{+}_{dd})W_k \\
&= \frac{1}{n(n-1)} \sum_{k=1}^{n} (W_k \sum_{s,d} (l^{+}_{sk} - l^{+}_{sd} - l^{+}_{dk} + l^{+}_{dd})) \\
&= \frac{1}{n(n-1)} \sum_{k=1}^{n} (W_k \times \frac{b_k}{W_k}) \\
&= \frac{1}{n(n-1)} \sum_{k=1}^{n} b_k \\
&= \frac{B}{n-1}
\end{aligned} \tag{4.31}$$

On the other hand, since $\frac{b_k}{W_k} = \frac{1}{2}\tau$ (equation 4.17), we have:

$$\begin{aligned} B &= \frac{\sum_k b_k}{n} = \frac{\sum_k \frac{1}{2}\tau W_k}{n} = \frac{1}{2}\tau\frac{\sum_k W_k}{n} \\ B &= \frac{1}{2}\bar{W}\tau \end{aligned}$$

Therefore using equation 4.31 we have:

$$\begin{aligned} T &= \frac{B}{n-1} = \frac{1}{2(n-1)}\bar{W}\tau \\ T &= \frac{n}{2}\bar{W}\hat{\tau} \end{aligned}$$

This completes the proof of lemma 4.5.2.

∎

Lemma 4.5.2 reveals that the average hop length of a random-walk is proportional to the product of average network criticality and average node weights. If we fix the total weight of a network at a budget C, then the average hop length of a walk would be proportional to the average network criticality, therefore, the average network criticality can quantify the average hop count of a random-walk.

4.5.3 Network Criticality and Average Betweenness Sensitivity

Another interpretation for network criticality is based on the betweenness of different network links. Since $\tau = \frac{b_{ij}}{w_{ij}}$, we have for $w_{ij} > 0$, and $w_{uv} > 0$

$$\begin{aligned} \frac{\partial \tau}{\partial w_{uv}} &= \frac{w_{ij}\frac{\partial b_{ij}}{\partial w_{uv}} - b_{ij}\frac{\partial w_{ij}}{\partial w_{uv}}}{w_{ij}^2} \\ &= \frac{1}{w_{ij}}\frac{\partial b_{ij}}{\partial w_{uv}} - \delta_{iu}\delta_{jv}\frac{b_{ij}}{w_{ij}^2} \\ &= \frac{1}{w_{ij}}\frac{\partial b_{ij}}{\partial w_{uv}} - \frac{\tau}{w_{ij}}\delta_{iu}\delta_{jv} \end{aligned}$$

Now we consider the special case where $(u, v) = (i, j)$.

$$\begin{aligned} \frac{\partial \tau}{\partial w_{ij}} &= \frac{1}{w_{ij}} \frac{\partial b_{ij}}{\partial w_{ij}} - \frac{\tau}{w_{ij}} \\ w_{ij} \frac{\partial \tau}{\partial w_{ij}} &= \frac{\partial b_{ij}}{\partial w_{ij}} - \tau \end{aligned} \tag{4.32}$$

By adding the results of equation 4.32 for different links of the network one can see:

$$\sum_{(i,j)\in E} w_{ij} \frac{\partial \tau}{\partial w_{ij}} = \sum_{(i,j)\in E} \frac{\partial b_{ij}}{\partial w_{ij}} - m\tau \tag{4.33}$$

Combining equation 4.33 and Lemma 4.4.3 results in:

$$\begin{aligned} \sum_{(i,j)\in E} \frac{\partial b_{ij}}{\partial w_{ij}} &= (m-1)\tau \\ \tau &= \frac{1}{m-1} \sum_{(i,j)\in E} \frac{\partial b_{ij}}{\partial w_{ij}} \end{aligned} \tag{4.34}$$

where m is the number of links of the network.

Observation 4.5.3 *According to equation 4.34 network criticality τ can be interpreted as the average of link betweenness derivatives or sensitivities with respect to link weight.*

Equation 4.34 suggests an effective approach to design routing and flow assignment algorithms. If we can estimate the variation of each link betweenness with respect to its weight (i.e. $\frac{\partial b_{ij}}{\partial w_{ij}}$), then we can use this variation as a cost to develop routing strategies to find min-cost paths. This idea will be developed in chapter 5.

4.5.4 Network Criticality and Algebraic Connectivity

Fiedler [94] defined algebraic connectivity as the first non-zero eigenvalue (λ_2) of the Laplacian matrix of a connected graph (recall from appendix A that the first eigenvalue of Laplacian matrix for a connected graph is zero). Algebraic connectivity is a lower bound for node connectivity and link connectivity. Therefore, *the further λ_2 is from zero, the higher the node and link connectivity of a graph.*

We now establish lower and upper bounds for network criticality based on algebraic connectivity.

Theorem 4.5.4 *Normalized network criticality satisfies the following bounds in terms of algebraic connectivity:* $\frac{2}{(n-1)\lambda_2} \leq \hat{\tau} \leq \frac{2}{\lambda_2}$.

Proof Lemma 4.4.2 can be used to obtain spectral bounds for network criticality. Since λ_2 is the smallest non-zero eigenvalue of graph Laplacian and all the eigenvalues are non-negative, we have:

$$\begin{aligned} \hat{\tau} &= \frac{2}{n-1}\sum_{i=2}^{n}\frac{1}{\lambda_i} \\ \hat{\tau} &\leq \frac{2}{n-1}\times\frac{n-1}{\lambda_2} \\ \hat{\tau} &\leq \frac{2}{\lambda_2} \end{aligned}$$

This establishes the upper bound for normalized network criticality. To get the lower bound we observe that:

$$\begin{aligned} \hat{\tau} &= \frac{2}{n-1}\sum_{i=2}^{n}\frac{1}{\lambda_i} \\ \hat{\tau} &\geq \frac{2}{n-1}\frac{1}{\lambda_2} \\ \hat{\tau} &\geq \frac{2}{(n-1)\lambda_2} \end{aligned}$$

combining these two inequalities results in:

$$\frac{2}{(n-1)\lambda_2} \leq \hat{\tau} \leq \frac{2}{\lambda_2}$$
$$\frac{2}{n-1} \leq \hat{\tau}\lambda_2 \leq 2$$

This completes the proof of theorem 4.5.4.

∎

Theorem 4.5.4 shows the relationship between network criticality and connectivity. Since normalized network criticality is upper bounded by the reciprocal of algebraic

connectivity, improvement of connectivity (increasing λ_2) improves the robustness as well (decreasing the upper bound of $\hat{\tau}$), but it is important to note that increasing connectivity at the same time decreases the lower bound of network criticality, which in turn causes more variance in network criticality. In other words, we can't uniformly improve the robustness of a network just by increasing the connectivity. Theorem 4.5.4 implies an effect similar to *Heisenberg Uncertainty Principle*. In quantum physics, the Heisenberg uncertainty principle states that the values of certain pairs of conjugate variables (position and momentum, for instance) cannot both be known with arbitrary precision. That is, the more precisely one variable is known, the less precisely the other is known. It appears that pair $(\hat{\tau}, \lambda_2)$ is another example of these conjugate pairs.

4.6 Calculation of Network Criticality for Some Graphs

To get a better understanding of the concept of criticality, we obtain exact expressions for network criticality of some well-structured network topologies.

4.6.1 Simple Link Network (K_2)

Here we calculate τ for a simple connected graph with two nodes and one link (Fig. 4.2). The weight and Laplacian matrices are:

$$W = \begin{pmatrix} 0 & a \\ a & 0 \end{pmatrix} \Rightarrow L = \begin{pmatrix} a & -a \\ -a & a \end{pmatrix} = a \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

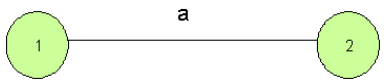


Figure 4.2: Simple Link Network

In order to find the Moore-Penrose inverse of Laplacian matrix L^+ we assume $L^+ = \begin{pmatrix} x & y \\ y & x \end{pmatrix}$, and use known matrix formula $LL^+L = L$ [93].

$$LL^+L = L$$

$$a^2\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x & y \\ y & x \end{pmatrix}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = a\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$a\begin{pmatrix} 2(x-y) & -2(x-y) \\ -2(x-y) & 2(x-y) \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$\Rightarrow \quad 2a(x-y) = 1$$

On the other hand the row sum of pseudo-Laplacian is zero (property of Laplacian matrix). Hence:

$$\begin{cases} 2a(x-y) = 1 \\ x + y = 0 \end{cases}$$

$$\Rightarrow \quad x = \tfrac{1}{4a}, \quad y = -\tfrac{1}{4a} \quad \Rightarrow \quad L^+ = \begin{pmatrix} \frac{1}{4a} & -\frac{1}{4a} \\ -\frac{1}{4a} & \frac{1}{4a} \end{pmatrix}$$

$$\tau_{12} = \tau_{21} = l^+_{11} + l^+_{22} - 2l^+_{12} = \tfrac{1}{a}$$

$$\tau(K_2) = \tau_{12} + \tau_{21} = \tfrac{2}{a}$$

$$\hat{\tau}(K_2) = \tfrac{1}{a}$$

This result shows that the average network criticality of K_2 is equal to the reciprocal of its link weight. The more the weight of the link, the less the criticality of the network. Theoretically, network criticality of K_2 approaches zero when the link weight

approaches ∞. This implies that the link weight can be interpreted as capacity. Indeed any beneficial QoS parameter can be incorporated into the definition of the link weight.

It is clear that increasing the weight improves the robustness of the network. The worst case happens when the link weight approaches zero ($a \to 0$). In this case network criticality approaches infinity ($\tau \to \infty$). This is the most critical situation where the connectivity of the network is violated. Connectivity in this context does not necessarily mean the physical connectivity, we consider a network K_2 disconnected if its transport service is interrupted (for example due to the link congestion).

4.6.2 Complete Graph on 3 Nodes (K_3)

We investigate τ for a complete graph on 3 nodes as shown in Fig. 4.3. We use equation 4.7 to obtain Moore-Penrose inverse of Laplacian L.

$$W = \begin{pmatrix} 0 & a & b \\ a & 0 & c \\ b & c & 0 \end{pmatrix} \Rightarrow L = \begin{pmatrix} a+b & -a & -b \\ -a & a+c & -c \\ -b & -c & b+c \end{pmatrix}$$

$$A = L(3|3) = \begin{pmatrix} a+b & -a \\ -a & a+c \end{pmatrix}$$

$$A^+ = L^+(3|3) = L^{-1}(3|3) = \frac{1}{ab+bc+ca} \begin{pmatrix} a+c & a \\ a & a+b \end{pmatrix}$$

Now we use the iterative formula for A^+ (equations 4.7 and 4.2). We let $z = \begin{pmatrix} -b \\ -c \end{pmatrix}$.

Step 1:

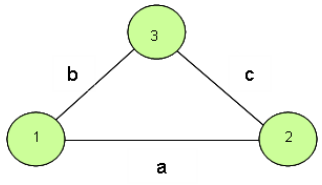


Figure 4.3: Complete Graph on 3 Nodes

$$
\begin{aligned}
A^+z &= \frac{1}{ab+bc+ca}\begin{pmatrix} a+c & a \\ a & a+b \end{pmatrix}\begin{pmatrix} -b \\ -c \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} = -\overrightarrow{1_2} \\
1 + z^*(AA^*)^+z &= 1 + (A^+z)^t(A^+z) = 1 + 2 = 3 \\
\zeta &= \frac{1}{3}z^*A^{*+}A^+ = \frac{1}{3}(A^+z)^tA^+ = -\frac{1}{3}\begin{pmatrix} 1 & 1 \end{pmatrix}\frac{1}{ab+bc+ca}\begin{pmatrix} a+c & a \\ a & a+b \end{pmatrix} \\
\zeta &= -\frac{1}{3(ab+bc+ca)}\begin{pmatrix} 2a+c & 2a+b \end{pmatrix} \\
A^+ - A^+z\zeta &= \frac{1}{ab+bc+ca}\begin{pmatrix} a+c & a \\ a & a+b \end{pmatrix} - \frac{1}{3(ab+bc+ca)}\begin{pmatrix} 1 \\ 1 \end{pmatrix}\begin{pmatrix} 2a+c & 2a+b \end{pmatrix} \\
&= \frac{1}{3(ab+bc+ca)}\begin{pmatrix} a+2c & a-b \\ a-c & a+2b \end{pmatrix} \\
\begin{pmatrix} A & z \end{pmatrix}^+ &= \frac{1}{3(ab+bc+ca)}\begin{pmatrix} a+2c & a-b \\ a-c & a+2b \\ -2a-c & -2a-b \end{pmatrix}
\end{aligned}
$$

Step 2:

Now we let $A = \begin{pmatrix} a+b & -a \\ -a & a+c \\ -b & -c \end{pmatrix}$ and $z = \begin{pmatrix} -b \\ -c \\ b+c \end{pmatrix}$.

Then we can apply second recursion on equation 4.7 to obtain final L^+. We repeat step

one with new values of A and z.

$$
\begin{aligned}
A^{+} &= \frac{1}{3(ab+bc+ca)}\begin{pmatrix} a+2c & a-c & -2a-c \\ a-b & a+2b & -2a-b \end{pmatrix} \\
A^{+}z &= \frac{1}{3(ab+bc+ca)}\begin{pmatrix} -ab-2bc-ac+c^2-2ab-2ac-bc-c^2 \\ -ab+b^2-ac-2bc-2ab-2ac-b^2-bc \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \\
1+z^{*}(AA^{*})^{+}z &= 1+(A^{+}z)^{t}(A^{+}z) = 1+2 = 3 \\
\zeta &= -\frac{1}{3}\begin{pmatrix} -1 \\ -1 \end{pmatrix}\frac{1}{3(ab+bc+ca)}\begin{pmatrix} a+2c & a-c & -2a-c \\ a-b & a+2b & -2a-b \end{pmatrix} \\
\zeta &= -\frac{1}{9(ab+bc+ca)}\begin{pmatrix} 2a+2c-b & 2a+2b-c & -4a-b-c \end{pmatrix} \\
A^{+}-A^{+}z\zeta &= \frac{1}{9(ab+bc+ca)}\begin{pmatrix} a+b+4c & a-2b-2c & -2a+b-2c \\ a-2b-2c & a+4b+c & -2a-2b+c \end{pmatrix}
\end{aligned}
$$

Now using equation $L^{+} = \begin{pmatrix} A & z \end{pmatrix}^{+}$ we can find L^{+}:

$$
L^{+} = \frac{1}{9(ab+bc+ca)}\begin{pmatrix} a+b+4c & a-2b-2c & -2a+b-2c \\ a-2b-2c & a+4b+c & -2a-2b+c \\ -2a-2c+b & -2a-2b+c & 4a+b+c \end{pmatrix} \tag{4.35}
$$

From equation 4.35 one can find the criticality of any two nodes as well as the network criticality.

$$
\tau_{12} = u_{12}^{t}L^{+}u_{12} = \frac{b+c}{ab+bc+ca} \tag{4.36}
$$

$$
\tau_{13} = u_{13}^{t}L^{+}u_{13} = \frac{a+c}{ab+bc+ca} \tag{4.37}
$$

$$
\tau_{23} = u_{23}^{t}L^{+}u_{23} = \frac{a+b}{ab+bc+ca} \tag{4.38}
$$

$$
\tau = \sum_{i\neq j}\tau_{ij} = 4\frac{a+b+c}{ab+bc+ca} \tag{4.39}
$$

$$
\hat{\tau} = \frac{1}{n(n-1)}\tau = \frac{2(a+b+c)}{3(ab+bc+ca)} \tag{4.40}
$$

Equations 4.36, 4.37, 4.38 show a similar pattern. The denominator of $\tau_{12}, \tau_{13}, \tau_{23}$ are equal to the cofactor of Laplacian matrix L, and the numerator of τ_{ij} (for $\tau_{12}, \tau_{13}, \tau_{23}$)

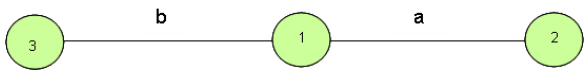


Figure 4.4: A Path of Length 2 on 3 nodes

appears to be the partial derivative of the cofactor with respect to the weight of link (i, j). In order to get a better understanding of these results, we consider some special cases.

4.6.2.1 Special Case: One link is shut down (A Linear Network on 3 nodes: P_3)

If one of the links in a K_3 graph is disconnected, the topology is converted to a path of length 2 on 3 nodes (P_3) as shown in Fig. 4.4. Let link between nodes 2 and 3 in graph of Fig. 4.3 be disconnected. This situation can be reached if the weight of the link (2,3) goes to zero ($c \to 0$). Applying $c \to 0$ to equations 4.36, 4.37, 4.38, and 4.39 provides one with criticality measures of P_3.

$$\begin{aligned}
\tau_{12}(P_3) &= \tau_{12}(K_3)|_{c\to 0} = \frac{1}{a} \\
\tau_{13}(P_3) &= \tau_{13}(K_3)|_{c\to 0} = \frac{1}{b} \\
\tau_{23}(P_3) &= \tau_{23}(K_3)|_{c\to 0} = \frac{a+b}{ab} = \frac{1}{a} + \frac{1}{b} \qquad (4.41) \\
\tau(P_3) &= \tau(K_3)|_{c\to 0} = 4\frac{a+b}{ab} = \frac{4}{a} + \frac{4}{b} \qquad (4.42) \\
\hat{\tau}(P_3) &= \frac{1}{3\times 2}\tau(P_3) = \frac{2(a+b)}{3(ab)}
\end{aligned}$$

Equation 4.41 shows that the criticality of node pair $2-3$ ($s = 3, d = 2$ or $s = 2, d = 3$) is the sum of the criticality of its path links. In addition, the network criticality of P_3 can be written as $\tau(P_3) = \frac{n_{12}}{w_{12}} + \frac{n_{13}}{w_{13}}$, where n_{ij} denotes the number of times that link (i, j) is in the path connecting one source to a destination. For instance $n_{12} = 4$, because there are four paths containing link $(1, 2)$. These paths are: $1 \to 2, 1 \to 2 \to 3, 2 \to 1, 3 \to 2 \to 1$.

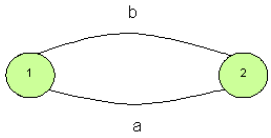


Figure 4.5: Graph R_2: Two Parallel Links

4.6.2.2 Special Case: One link has unlimited capacity (Ring on 2 Nodes: R_2)

Another special case for K_3 is when capacity of one link is unlimited. Let $c \to \infty$, then the complete graph is converted to a network with 2 nodes and 2 parallel links (Fig. 4.5) or a ring of length 2. We call this graph R_2. Applying $c \to \infty$ to equations 4.36, 4.37, 4.38, and 4.39 provides one with criticality measures of graph R_2.

$$\tau_{12}(R_2) = \tau_{12}(K_3)|_{c\to\infty} = \frac{1}{a+b} \tag{4.43}$$

$$\tau_{13}(R_2) = \tau_{13}(K_3)|_{c\to\infty} = \frac{1}{a+b} \tag{4.44}$$

$$\tau_{23}(R_2) = \tau_{23}(K_3)|_{c\to\infty} = 0 \tag{4.45}$$

$$\tau(R_2) = \tau(K_3)|_{c\to\infty} = \frac{4}{a+b} \tag{4.46}$$

$$\hat{\tau}(R_2) = \frac{1}{2\times(2-1)}\tau(R_2) = \frac{2}{a+b} \tag{4.47}$$

Equations 4.43, 4.44 show that in the context criticality, two parallel links with weights a and b can be replaced by one link whose weight is the sum of the weights of original parallel links i.e. $a + b$. Equation 4.45 verifies that two end-nodes of a link with infinite weight can be merged. Finally, equation 4.46 shows that graph R_2 is disconnected only if both parallel links are disconnected, that is, with regard to topological connectivity, two parallel links are more robust comparing one link with higher weight (no matter what the link weights are). Note that this does not mean that R_2 (two parallel links topology) is always more robust than K_2 (one link network).

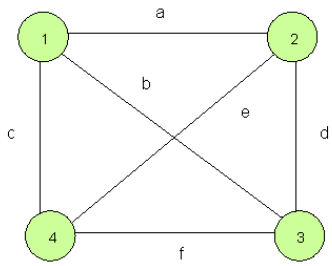


Figure 4.6: Complete Graph on 4 Nodes (K_4)

4.6.3 Complete Graph on 4 Nodes (K_4)

We investigate τ for a complete graph on 4 nodes as shown in Fig. 4.6. We omit the details of derivations as it is exactly similar to what we did for K_3 in §4.6.2

$$\begin{aligned}
\tau_{12} &= \frac{df + eb + ed + ef + bf + cb + cd + cf}{X} \\
\tau_{13} &= \frac{ce + cd + ca + ef + ed + df + af + ae}{X} \\
\tau_{14} &= \frac{ab + ad + af + db + df + eb + ed + ef}{X} \\
\tau_{23} &= \frac{af + bf + cb + cf + ae + eb + ca + ce}{X} \\
\tau_{24} &= \frac{ab + ad + af + db + bf + cb + cd + cf}{X} \\
\tau_{34} &= \frac{ad + ae + ab + db + eb + ca + cd + ce}{X} \\
\tau &= \frac{2X_1}{X} \qquad \hat{\tau} = \frac{X_1}{6X}
\end{aligned}$$

where

$$\begin{aligned}
X_1 &= 3ca + 3ce + 4eb + 3df + 4af + 3bf + 4cd + 3cb \\
&+ 3cf + 3ae + 3db + 3ad + 3ab + 3ef + 3ed \\
X &= cef + ced + ceb + cdf + cdb + caf + cad + cab \\
&+ bef + bed + bdf + baf + aef + aed + aeb + adf
\end{aligned}$$

One can notice that the denominator of τ is the cofactor of the graph Laplacian.

4.6.3.1 Special Case: Ring on 4 Nodes (R_4)

One can find network criticality of a ring with 4 nodes by removing links $(1,3)$ and $(2,4)$ from topology of K_4. This means that link weights b and e should be zero. R_4 topology is shown in Fig. 4.7. Criticality of different $S-D$ pairs and total criticality for R_4 are listed here.

$$
\begin{aligned}
\tau_{12}(R_4) &= \tau_{12}(K4)|_{b,c\to 0} = \frac{cf+cd+df}{cdf+caf+cad+adf} \\
\tau_{13}(R_4) &= \tau_{13}(K4)|_{b,c\to 0} = \frac{af+df+ca+cd}{cdf+caf+cad+adf} \\
\tau_{14}(R_4) &= \tau_{14}(K4)|_{b,c\to 0} = \frac{ad+af+df}{cdf+caf+cad+adf} \\
\tau_{23}(R_4) &= \tau_{23}(K4)|_{b,c\to 0} = \frac{af+cf+ca}{cdf+caf+cad+adf} \\
\tau_{24}(R_4) &= \tau_{24}(K4)|_{b,c\to 0} = \frac{ad+af+cd+cf}{cdf+caf+cad+adf} \\
\tau_{34}(R_4) &= \tau_{34}(K4)|_{b,c\to 0} = \frac{ad+ca+cd}{cdf+caf+cad+adf} \\
\tau(R_4) &= \tau(K_4)|_{b,c\to 0} = \frac{2(4cd+3cf+3ca+3ad+3df+4af)}{cdf+caf+cad+adf} \\
\hat{\tau}(R_4) &= \frac{4cd+3cf+3ca+3ad+3df+4af}{6(cdf+caf+cad+adf)}
\end{aligned}
$$

These results show that end-to-end criticality and network criticality are polynomial fractions, where the numerator and the denominator are polynomial functions of link weights and the denominator is always equal to the cofactor of the Laplacian matrix.

4.6.3.2 Special Case: Linear Network on 4 Nodes (P_4)

One can find network criticality of a path on 4 nodes (length of 3) by removing one link from ring topology R_4. We remove link $(2,3)$ from R_4. The result is P_4 which is shown

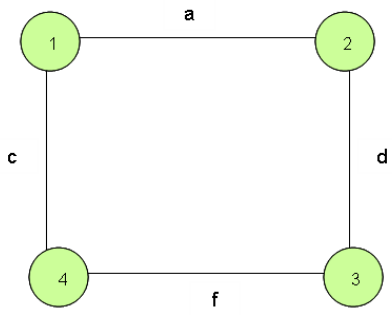


Figure 4.7: Ring on 4 Nodes (R_4)

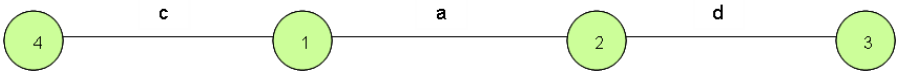


Figure 4.8: Path on 4 Nodes (P_4)

in Fig. 4.8. Criticality between different $S-D$ pairs are calculated in the following:

$$\begin{aligned}
\tau_{12}(P_4) &= \tau_{12}(R_4)|_{f\to 0} = \frac{1}{a} \\
\tau_{13}(P_4) &= \tau_{13}(R_4)|_{f\to 0} = \frac{1}{a}+\frac{1}{d} \qquad (4.48)\\
\tau_{14}(P_4) &= \tau_{14}(R_4)|_{f\to 0} = \frac{1}{c} \\
\tau_{23}(P_4) &= \tau_{23}(R_4)|_{f\to 0} = \frac{1}{d} \\
\tau_{24}(P_4) &= \tau_{24}(R_4)|_{f\to 0} = \frac{1}{a}+\frac{1}{c} \qquad (4.49)\\
\tau_{34}(P_4) &= \tau_{34}(R_4)|_{f\to 0} = \frac{1}{a}+\frac{1}{c}+\frac{1}{d} \qquad (4.50)\\
\tau(P_4) &= \tau(R_4)|_{f\to 0} = 2(\frac{4}{a}+\frac{3}{c}+\frac{3}{d}) \qquad (4.51)
\end{aligned}$$

Equations 4.41, 4.48, 4.49, 4.50 show that the criticality of source-destination pair $s-d$ is the sum of the criticality of its links. In addition, according to equation 4.51 the network criticality of $P4$ can be written as $\tau(P_4) = \frac{n_a}{a} + \frac{n_c}{c} + \frac{n_d}{d}$, where n_a shows the number of times that link with weight a is in the path connecting one source to a destination.

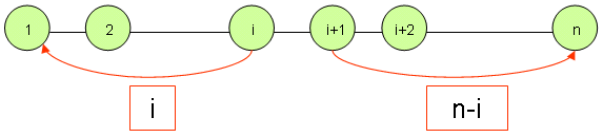


Figure 4.9: Network Criticality for (P_n)

4.6.4 Criticality for a General Linear Graph on n Nodes (P_n)

Equations 4.41, 4.48, 4.49, 4.50 suggest a general formula for obtaining the criticality of any source-destination pair $s-d$ on a path. Criticality of any node pair $s-d$ in a linear graph is the sum of the reciprocal of all link weights between s and d. Note that there is only one path between any source-destination pair as our graph is a path by itself.

Equations 4.42 and 4.51 also suggest a general formula for calculating network criticality of a path. The criticality of P_n can be written as $\tau(P_n) = \sum_{(i,j)\in E} \frac{n_{ij}}{w_{ij}}$, where n_{ij} shows the number of times that link (i, j) is in an arbitrary path connecting one source node to a destination node.

Suppose nodes of a path P_n are numbered from 1 to n. Then the number of paths containing link $(i, i+1)$ is equal to $n_{ij} = 2i(n-i)$ (Fig. 4.9). The coefficient 2 accounts for the paths in reverse direction.

$$\tau(P_n) = \sum_{i\in N} \frac{2i(n-i)}{w_{i,i+1}} \tag{4.52}$$

4.6.5 Criticality for a General Tree

Since a tree is a graph without cycle, there is only one path between every source-destination pair. The criticality of this single path can be found by adding the criticality of its links (reciprocal of link weight). The criticality of the tree is the sum of the criticality of all such paths. If n_{ij} denotes the number of times all such paths include

link (i, j), the criticality of the tree can be written as:

$$\tau = \sum_{(i,j)\in E} \frac{n_{ij}}{w_{ij}} \tag{4.53}$$

Chapter 5

Design of Robust Networks

In this chapter we elaborate on the use of network criticality as a metric to quantify robustness of a network. We show that network criticality is a strictly convex function of network weight matrix and we develop an optimization problem to minimize the criticality. The optimization then leads to guidelines for network design problem and network traffic management either online of offline. Essentially, the idea is to find the difference between the solution of primal and dual convex optimization problem (duality gap). For a strictly convex function, the duality gap has to be zero for the optimal solution. In sub-optimal cases, the duality gap is always greater than zero. The algorithms are built based on the idea of minimizing the duality gap to the extent possible. In the following sections, we try to shed more light on this approach.

5.1 Why is Network Criticality so Important?

In chapter 4 we have shown some important interpretations of network criticality. We saw that the average cost, average walk-length, and average link betweenness sensitivity are proportional to the network criticality. In this section we study another important aspect of network criticality. In chapter 3 we investigated the behavior of

the betweenness in the presence of external input traffic rate. We showed that for a Jackson network, the maximization of carried load is equal to minimizing the average of *NCI* when the network is lightly loaded, or to minimizing the maximum of *NCI* when the network is heavily loaded (see theorem 3.6.2). In this section we extend those results to general networks and show that to maximize carried load, *one needs to minimize the network criticality*. We start by deriving a useful result about betweenness as a direct consequence of equation 4.17.

Corollary 5.1.1 *The normalized betweenness of node i in a graph is* $\pi_i = \frac{b_i}{\sum_{k=1}^{n} b_k}$, *where* π_i *is the stationary probability of node i in a Markov chain with transition probabilities* $p_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}$.

Proof Equation 4.17 can be used to simplify the normalized node betweenness.

$$\begin{aligned} \textit{Normalized } b_i &= \frac{b_i}{\sum_{k=1}^{n} b_k} \\ &= \frac{\frac{1}{2}\tau W_i}{\frac{1}{2}\sum_{k=1}^{n} \tau W_k} \\ &= \frac{W_i}{W} \\ &= \pi_i \end{aligned}$$

where $W_i = \sum_{j=1}^{n} w_{ij}$, $W = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}$. This completes the proof of corollary 5.1.1.

∎

Let λ be the average input rate of the network, and let the weight of each link be the capacity of the link $(i, j) = l$ (i.e. $w_{ij} = c_{ij} = c(l)$). Further, let x_k be the average load on node k and c_k be the capacity of node k. By applying Little's formula and using corollary 5.1.1 and equation 4.31 we have:

$$\begin{aligned} x_k &= \lambda \pi_k T \\ &= \lambda \frac{b_k}{\sum_i b_i} T \\ &= \lambda \frac{b_k}{nB} \frac{B}{n-1} \\ x_k &= \frac{\lambda}{n(n-1)} b_k \end{aligned} \tag{5.1}$$

Therefore, if we denote the maximum load with x_{max} and the capacity of the node that carries maximum load with c^* then we have:

$$\begin{aligned} x_{max} &\leq c^* \\ \frac{\lambda}{n(n-1)} b_{max} &\leq c^* \\ \lambda &\leq \frac{n(n-1)}{\frac{b_{max}}{c^*}} \\ \lambda &\leq \frac{n(n-1)}{\eta} \end{aligned}$$

Since we know $\eta = \frac{\tau}{2}$ (see definition 4.3.1), we will have:

$$\lambda \leq \frac{2}{\hat{\tau}} \tag{5.2}$$

We can summarize these results in theorems 5.1.2 and 5.1.3.

Theorem 5.1.2 *Consider a network topology G(N,E,W), and assume there is external expected input traffic rate of λ for the network. Let x_k be the expected load on node k, and b_k be the total betweenness of node k, then:*

$$x_k = \frac{\lambda}{n(n-1)} \times b_k$$

Theorem 5.1.3 *To maximize the carried load of a network, one needs to minimize the network criticality, where the link weight is defined as the link capacity:*

$$\max_W \lambda = \frac{2}{\min_W \hat{\tau}}$$

Proof The proof is a direct result of equation 5.2. According to this equation, the offered load λ is bounded to the reciprocal of network criticality. This means that to maximize λ one needs to minimize η (or network criticality).

∎

Observation 5.1.4 *Comparing theorem 5.1.3 and 3.6.2 one can notice that in general the optimization problem defined in theorem 5.1.3 is incorporating both cases of theorem 3.6.2.*

Theorem 5.1.3 reveals the importance of criticality in the study of networks. Reciprocal of the network criticality provides an upper bound for the maximum load that a network can carry. If input traffic goes beyond the limit of theorem 5.1.3, the network experiences congestion. In other words by minimizing the network criticality we delay the onset of congestion to the extent possible.

In the following sections we will study dynamic properties of network criticality and show that the minimization of τ is possible. Then we will derive useful directions to design networks with appropriate network criticality, or to design control algorithms to maintain network criticality as low as possible. The latter situation happens when the weight matrix changes dynamically, which in turn changes network criticality.

5.2 Convexity of Network Criticality

In this section we prove an important property of network criticality. We derive expressions for first, second, and in general n^{th} derivative of τ with respect to link weight w_{ij} and using the results we show that network criticality τ is a strictly convex function of link weights. We start by stating a well-known fact from graph-theory about generalized inverse matrix of a graph.

Proposition 5.2.1 *For a connected graph, matrix $(L + \frac{J}{n})^{-1}$ is invertible and the generalized inverse (Moore-Penrose inverse) of the Laplacian matrix is:*

$$L^{+} = (L + \frac{J}{n})^{-1} - \frac{J}{n} \tag{5.3}$$

where matrix J in this equation is an $n \times n$ matrix with all elements equal to 1.

Proof See [95]. ∎

In order to prove the convexity of network criticality we need the following lemmas.

Lemma 5.2.2 *Network criticality* ($\hat{\tau}$) *is a monotone decreasing function of weights. In addition, the first partial derivative of* τ *with respect to a link weight* w_{ij} *is:*

$$\frac{\partial \hat{\tau}}{\partial w_{ij}} = -\frac{2}{n-1}\|L_i^+ - L_j^+\|^2$$

where L_i^+ *denotes the* i^{th} *column of matrix* L^+.

Proof In chapter 4, we saw that the network criticality is a polynomial fraction, where the numerator and the denominator are polynomial functions of link weights. It follows that $\frac{\partial^n \tau}{\partial w_{ij}^n}$ exist except at the poles of τ. Using equations 4.20 and 5.3 we can calculate the first derivative of τ_{sd} for the weight of a typical link $l = (i, j)$.

$$\frac{\partial \tau_{sd}}{\partial w_{ij}} = u_{sd}^t \times \frac{\partial L^+}{\partial w_{ij}} \times u_{sd} \tag{5.4}$$

In order to calculate the derivative of matrix L^+ with respect to w_{ij} we use equation 5.3′: $\frac{\partial L^+}{\partial w_{ij}} = \frac{\partial (L+\frac{J}{n})^{-1}}{\partial w_{ij}}$. On the other hand:

$$\begin{gathered}
(L + \tfrac{J}{n})^{-1} \times (L + \tfrac{J}{n}) = I \\
\tfrac{\partial (L+\frac{J}{n})^{-1}}{\partial w_{ij}} \times (L + \tfrac{J}{n}) + (L + \tfrac{J}{n})^{-1} \times \tfrac{\partial (L+\frac{J}{n})}{\partial w_{ij}} = 0 \\
\tfrac{\partial (L+\frac{J}{n})^{-1}}{\partial w_{ij}} = -(L + \tfrac{J}{n})^{-1} \times \tfrac{\partial L}{\partial w_{ij}} \times (L + \tfrac{J}{n})^{-1} \\
\tfrac{\partial L^+}{\partial w_{ij}} = -(L + \tfrac{J}{n})^{-1} \times \tfrac{\partial L}{\partial w_{ij}} \times (L + \tfrac{J}{n})^{-1}
\end{gathered} \tag{5.5}$$

Replacing equation 5.5 in equation 5.4 gives:

$$\begin{aligned}
\frac{\partial \tau_{sd}}{\partial w_{ij}} &= u_{sd}^t \times \frac{\partial L+}{\partial w_{ij}} \times u_{sd} \\
&= -1 \times u_{sd}^t \times (L + \frac{J}{n})^{-1} \times \frac{\partial (L + \frac{J}{n})}{\partial w_{ij}} \times (L + \frac{J}{n})^{-1} \times u_{sd} \\
&= -1 \times u_{sd}^t \times (L + \frac{J}{n})^{-1} \times \frac{\partial L}{\partial w_{ij}} \times (L + \frac{J}{n})^{-1} \times u_{sd}
\end{aligned} \tag{5.6}$$

To obtain $\frac{\partial L}{\partial w_{ij}}$ we notice that w_{ij} appears in four elements of matrix L: l_{ij}, , l_{ji}, , l_{ii}, , l_{jj}, .

For these elements based on the definition of Laplacian ($L = D - W$) we have:

$$(\frac{\partial L}{\partial w_{ij}})_{pq} = \begin{cases} 1 & p = i,\ q = i \\ 1 & p = j,\ q = j \\ -1 & p = i,\ q = j \\ -1 & p = j,\ q = i \\ 0 & otherwise \end{cases} \tag{5.7}$$

Equation 5.7 can be written in matrix form as:

$$\frac{\partial L}{\partial w_{ij}} = u_{ij} \times u_{ij}^t \tag{5.8}$$

Equation 5.8 can also be found directly from an alternative definition of Laplacian for weighted graphs given in appendix A equation A.1. Combining equations 5.6 and 5.8,

$$\frac{\partial \tau_{sd}}{\partial w_{ij}} = -1 \times u_{sd}^t \times (L + \frac{J}{n})^{-1} \times u_{ij} \times u_{ij}^t \times (L + \frac{J}{n})^{-1} \times u_{sd} \tag{5.9}$$

Using the fact that $Ju_{ij} = u_{ij}^t J = 0$, one can easily verify that:

$$\begin{aligned} (L + \frac{J}{n})^{-1} u_{ij} &= ((L + \frac{J}{n})^{-1} - \frac{J}{n}) u_{ij} \\ &= L^+ u_{ij} = L_i^+ - L_j^+ \end{aligned} \tag{5.10}$$

where L_i^+ is the i^{th} column of L^+. Using equation 5.10 in 5.9 gives:

$$\begin{aligned} \frac{\partial \tau_{sd}}{\partial w_{ij}} &= -1 \times u_{sd}^t (L_i^+ - L_j^+)(L_i^+ - L_j^+)^t u_{sd} \\ &= -1 \times ((L_i^+ - L_j^+)^t u_{sd})^t (L_i^+ - L_j^+)^t u_{sd} \\ &= -1 \times ((l_{is}^+ - l_{id}^+) - (l_{js}^+ - l_{jd}^+))^2 \\ &= -1 \times ((l_{is}^+ - l_{js}^+)^2 + (l_{id}^+ - l_{jd}^+)^2 - 2(l_{is}^+ - l_{js}^+)(l_{id}^+ - l_{jd}^+)) \end{aligned} \tag{5.11}$$

Now we are ready to obtain the derivative of τ.

$$\begin{aligned}
\frac{\partial \tau}{\partial w_{ij}} &= \sum_d \sum_s \frac{\partial \tau_{sd}}{\partial w_{ij}} \\
&= -1 \times (\sum_d \sum_s (l^+_{is} - l^+_{js})^2 + \sum_d \sum_s (l^+_{id} - l^+_{jd})^2 - \\
&\quad 2 \sum_d \sum_s (l^+_{is} - l^+_{js})(l^+_{id} - l^+_{jd})) \\
&= -1 \times (n\|L^+ u_{ij}\|^2 + n\|L^+ u_{ij}\|^2 - \\
&\quad 2 \sum_d (l^+_{id} - l^+_{jd}) \sum_s (l^+_{is} - l^+_{js})) \\
&= -2n\|L^+ u_{ij}\|^2 \\
\frac{\partial \tau}{\partial w_{ij}} &= -2n\|L^+_i - L^+_j\|^2 \qquad (5.12)
\end{aligned}$$

Note that $\|L^+_i - L^+_j\|^2 > 0$ because if $\|L^+_i - L^+_j\|^2 = 0$ then $rank(L^+) < (n-1)$ and this violates the connectivity of our graph. Therefore 5.12 shows that the derivative of τ is always negative. This completes the proof of lemma 5.2.2.

∎

Lemma 5.2.3 *The first partial derivative of τ_{ij} with respect to w_{ij} satisfies the following partial differential equation.*

$$\frac{\partial \tau_{ij}}{\partial w_{ij}} = -\tau^2_{ij}$$

Proof We use equation 5.11 to find the partial derivative of τ_{ij}

$$\begin{aligned}
\frac{\partial \tau_{ij}}{\partial w_{ij}} &= -1 \times u^t_{ij}(L^+_i - L^+_j)(L^+_i - L^+_j)^t u_{ij} \\
&= -u^t_{ij} L^+ u_{ij} u^t_{ij} L^+ u_{ij} \\
&= -\tau^2_{ij} \qquad (5.13)
\end{aligned}$$

This completes the proof of lemma 5.2.3.

Now we find higher order derivatives of network criticality.

Lemma 5.2.4 *The n^{th} derivative of τ with respect to the link weights is:*

$$\frac{\partial^n \tau}{\partial {w_{ij}}^n} = n!(-\tau_{ij})^{n-1}\frac{\partial \tau}{\partial w_{ij}} \tag{5.14}$$

Proof We use induction. First we find the second partial derivative of τ with respect to w_{ij}.

$$\begin{aligned}
\frac{\partial \tau}{\partial w_{ij}} &= -2n\|L^+ u_{ij}\|^2 \\
&= -2nL^+ u_{ij} u_{ij}^t L^+ \\
\frac{\partial^2 \tau}{\partial {w_{ij}}^2} &= -2n\frac{\partial}{\partial w_{ij}}(L^+ u_{ij} u_{ij}^t L^+) \\
&= 2n(\frac{\partial L^+}{\partial w_{ij}} u_{ij} u_{ij}^t L^+ + L^+ u_{ij} u_{ij}^t u_{ij}\frac{\partial L^+}{\partial w_{ij}}) \\
&= -2n(-L^+ u_{ij} u_{ij}^t L^+ u_{ij} u_{ij}^t L^+ - L^+ u_{ij} u_{ij}^t L^+ u_{ij} u_{ij}^t L^+) \\
&= -2n \times (-2)\tau_{ij} L^+ u_{ij} u_{ij}^t L^+ \\
\frac{\partial^2 \tau}{\partial {w_{ij}}^2} &= -2\tau_{ij}\frac{\partial \tau}{\partial w_{ij}}
\end{aligned} \tag{5.15}$$

This proves that equation 5.14 is valid for $n = 2$. Now we assume that equation 5.14 is true for $n = k$, we prove that it will be true for $n = k + 1$ too.

$$\begin{aligned}
\frac{\partial^{k+1}\tau}{\partial w_{ij}^{k+1}} &= \frac{\partial}{\partial w_{ij}}\{k!(-\tau_{ij})^k \frac{\partial \tau}{\partial w_{ij}}\} \\
&= k!\{(k-1)(-\frac{\partial \tau_{ij}}{\partial w_{ij}})(-\tau_{ij})^{k-2}\frac{\partial \tau}{\partial w_{ij}} + (-\tau_{ij})^{k-1}\frac{\partial^2 \tau}{\partial w_{ij}^2}\} \\
&= k!(-\tau_{ij})^{k-2}\{-(k-1)\frac{\partial \tau_{ij}}{\partial w_{ij}}\frac{\partial \tau}{\partial w_{ij}} + (-\tau_{ij})(-2\tau_{ij})\frac{\partial \tau}{\partial w_{ij}}\} \\
&= k!(-\tau_{ij})^{k-2}\frac{\partial \tau}{\partial w_{ij}}\{-(k-1)\frac{\partial \tau_{ij}}{\partial w_{ij}} + 2\tau_{ij}^2\}
\end{aligned} \tag{5.16}$$

We use lemma 5.2.3 in equation 5.16:

$$\begin{aligned}
\frac{\partial^{k+1}\tau}{\partial w_{ij}^{k+1}} &= k!(-\tau_{ij})^{k-2}\frac{\partial \tau}{\partial w_{ij}}\{-(k-1)(-\tau_{ij}^2) + 2\tau_{ij}^2\} \\
&= k!(-\tau_{ij})^{k-2}\frac{\partial \tau}{\partial w_{ij}}\{(k+1)\tau_{ij}^2\} \\
&= (k+1)!(-\tau_{ij})^k \frac{\partial \tau}{\partial w_{ij}}
\end{aligned}$$

This completes the proof of lemma 5.2.4.

∎

Theorem 5.2.5 *τ is a strictly convex function of the graph weights.*

Proof: It is sufficient to show that the first derivative of τ is always negative and it's second derivative is always non-negative. Lemma 5.2.2 verifies that the first partial derivative of τ with respect to w_{ij} is negative and lemma 5.2.4 shows that the second derivative of τ is non-negative which means that τ is a strict convex function on weight set. This completes the proof of theorem 5.2.5.

∎

5.3 Convex Optimization Problem for Network Criticality

Theorem 5.2.5 has some direct consequences.

Observation 5.3.1 *The problem of finding the graph weights to optimize a specific metric is a convex optimization problem and all the related literature can be used to solve it.*

Observation 5.3.2 *Due to the fact that τ is a strictly convex function of the weights, an optimization problem with some constraints has a unique solution.*

The ultimate goal is to come up with a method to minimize the criticality. Our results show that minimizing node/link criticality is equivalent to minimizing the function τ. Hence, we consider the minimization of τ under some constraints. We assume that there is a cost $z_{ij} = z_{ji}$ related to each link weight w_{ij}. A reasonable constraint is to assume that we have a maximum total budget C. This constraint is motivated by observation 4.5.1.

Theorem 5.3.3 *Consider the following optimization problem on graph $G(N, E, W)$:*

$$\begin{aligned} & \textit{Minimize} \quad \tau \\ \textit{Subject to} \quad & \textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} = C \quad , C \textit{ is fixed} \\ & w_{ij} \geq 0 \end{aligned} \tag{5.17}$$

For the optimal weight set, W^*, we have:

$$w^*_{ij}(C\frac{\partial \tau}{\partial w_{ij}} + z_{ij}\tau) = 0 \quad (i,j) \in E \tag{5.18}$$

Proof The cost constraint can be written as an inner product of costs and weights, so we can write

$$(Vec(Z).Vec(W^*))\tau = (\sum_{(i,j)\in E} w^*_{ij} z_{ij})\tau = C\tau \tag{5.19}$$

Combining lemma 4.4.3 and equation 5.19 one can see:

$$\begin{aligned} C\nabla\tau.Vec(W^*) + Vec(Z).Vec(W^*)\tau &= 0 \\ Vec(W^*).(C\nabla\tau + \tau Vec(Z)) &= 0 \\ w^*_{ij}(C\frac{\partial \tau}{\partial w_{ij}} + \tau z_{ij}) &= 0 \end{aligned} \tag{5.20}$$

This completes the proof of theorem 5.3.3.

∎

Observation 5.3.4 *Theorem 5.3.3 shows that the partial derivative of τ with respect to any optimal weight w^*_{ij} is proportional to the τ if the link weight is non-zero.*

In general, one can apply the condition of optimality [96, 97] on optimization problem 5.17 to get necessary condition for a weight vector to be optimal. Let W^* be the optimal weight matrix, and let W_t be another weight matrix satisfying the constraints of optimization problem 5.17, then according to the condition of optimality:

$$\nabla\tau.(Vec(W_t) - Vec(W^*)) \geq 0$$

Now, we choose W_t as follows:

$$W_t = [w_{uv}] = \begin{cases} \frac{C}{2z_{ij}} & if \;\; u = i \;\&\; v = j \\ \frac{C}{2z_{ij}} & if \;\; u = j \;\&\; v = i \\ 0 & otherwise \end{cases}$$

Clearly, W_t satisfies the constraints of optimization problem 5.17, therefore, using the condition of optimality and considering lemma 4.4.3 we have:

$$\begin{aligned} \nabla\tau.(Vec(W_t) - Vec(W^*)) &\geq 0 \\ \nabla\tau.Vec(W_t) - \nabla\tau.Vec(W^*) &\geq 0 \\ \frac{C}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}} + \tau &\geq 0 \quad \forall(i,j) \in E \\ \min_{(i,j)\in E} \frac{C}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}} + \tau &\geq 0 \end{aligned} \tag{5.21}$$

The constraints of optimization problem 5.17 and inequality 5.21 state necessary and sufficient conditions for the optimality of any weight matrix.

5.4 Design of a Robust Routing Algorithm for AutoNet

Before we go into further details of the optimization problem 5.17, we use equation 5.20 to derive an algorithm for robust routing. In chapter 4 we saw that network criticality can be viewed as the average of changes in link betweennesses with respect to the weights (see §4.5.3). Equation 4.34 suggests that we can use $\frac{\partial b_{ij}}{\partial w_{ij}}$ as link cost and try to design algorithms to run network flows in paths with less cost. Lets consider a network in optimal condition, i.e. satisfying equation 5.20. We can find a simple expression for $\frac{\partial b_{ij}}{\partial w_{ij}}$ (when the weight set is optimal) by combining equation 5.20 and

4.34.

$$\begin{aligned} b_{ij} &= \tau w_{ij} \\ \frac{\partial b_{ij}}{\partial w_{ij}} &= \frac{\partial \tau}{\partial w_{ij}} w_{ij} + \tau \\ \frac{\partial b_{ij}}{\partial w_{ij}} &= \tau(1 - \frac{z_{ij} w_{ij}}{C}) \end{aligned} \tag{5.22}$$

Equation 5.22 has a nice interpretation. For small weight costs ($\frac{z_{ij} w_{ij}}{C} \ll 1$) the variation of link betweenness with respect to the changes in link weight is near the network criticality. For links with higher cost, the betweenness sensitivity is smaller. Note that equation 5.22 is the expression for link betweenness sensitivity only if we have optimal weight setting, but in general, the weight set is not optimal and using equation 5.22 is not a right choice (for example if we use this equation as the link cost for Dijkstra's algorithm, the optimal path using Dijkstra's method will be the same as min-hop shortest path method, which is clearly not an improvement for us). Instead we need to directly obtain the link betweenness sensitivity, one simple method to use the approximation:

$$\frac{\partial b_{ij}}{\partial w_{ij}}|_{t=k} \approx \frac{b_{ij}(k) - b_{ij}(k-1)}{w_{ij}(k) - w_{ij}(k-1)}$$

where $t = k$ shows k^{th} step. In next section we will show how to use link betweenness sensitivity to design an improved version of PCR algorithm.

5.4.1 Random-Walk Path Criticality Routing Algorithm (RW-PCR)

The idea of RW-PCR is simple. We label each and every link of the graph with its $\frac{\partial b_{ij}}{\partial w_{ij}}$ as the cost and use Dijkstra's algorithm to obtain the shortest path(s) from a source s to a destination d using the assigned cost for the links. Bear in mind that this cost is not the same as z_{ij}. In fact the cost has the same role as LCI in our heuristic PCR algorithm which is described in chapter 3 (§3.3).

Definition 5.4.1 *We call $\frac{\partial b_{ij}}{\partial w_{ij}}$ as Random-Walk Link Criticality Index (LCI_{RW}).*

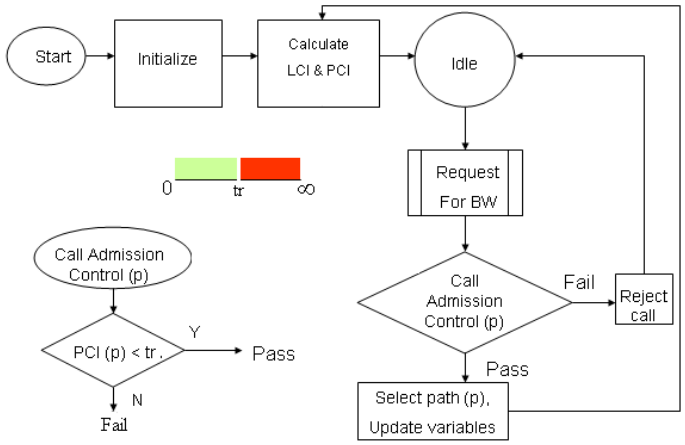


Figure 5.1: Flowchart of RW-PCR Algorithm

Definition 5.4.2 *We define the maximum of the LCI_{RW} of the links in a path as Random-Walk Path Criticality Index (PCI_{RW}). This is motivated by equation 5.2. For a path $p = uu_1u_2...u_kv$:*

$$PCI_{RW}(p) = \max(LCI_{RW}(u, u_1), ..., LCI_{RW}(u_{k-1}, u_k), LCI_{RW}(u_k, v))$$

where $LCI_{RW}(x, y)$ denotes the LCI_{RW} of link (x, y).

When a demand for source-destination pair $s-d$ arrives, the shortest path obtained in this way would be considered as a candidate to be assigned to the demand. A simple call-admission control is applied here by considering a threshold tr for the criticality of the path. If the PCI_{RW} is more than this threshold, then the flow would be considered too risky for the network and be rejected (blocked), otherwise the path is used as the route and the demand flow is assigned to this path. The available bandwidth of all the links on this path is updated and the LCI's are also modified accordingly. In case of having more than one shortest path, the path with least PCI_{RW} will be chosen. A simple flowchart of RW-PCR algorithm is shown in Fig. 5.1.

5.4.1.1 Time Complexity of RW-PCR Argorithm

To estimate the time complexity of the algorithm, we note that the running time to get the Moore-Penrose inverse is $O(mn^{\frac{1}{2}})$ [93], where m and n are the number of links and nodes in the graph respectively. The main part of the RW-PCR can be obtained in $O(nlog(n))$ as it is just a shortest path algorithm with link costs. Therefore, the complexity of the algorithm would be $O(mn^{\frac{3}{2}}log(n))$.

5.4.1.2 Evaluation of RW-PCR

In order to investigate the effectiveness of our RW-PCR algorithm, we run a set of simulations similar to those we conducted for evaluation of PCR in §3.4.5 on the same network (Fig. 3.10). We apply RW-PCR to create LSPs (Label Switch Path) assuming that MPLS is used in the network to create the paths.

In the first experiment the requests for LSPs arrive at random and stay forever (no departures). In our tests the bandwidth requests for paths (LSPs) are taken to be uniformly distributed between 1 to 3 units. In Fig. 5.2 we show the number of rejected calls for this case and compare the performance to that of original deterministic PCR , shortest path (SP), and widest shortest path (WSP). The test is performed 20 times and each time with 2000 path requests. We measured the number of blocked requests. Fig. 5.2 shows that RW-PCR algorithm has the best performance and the performance of PCR and RW-PCR algorithms are much better than WSP and SP.

In another experiment we examined the behavior of the algorithms in the presence of dynamic traffic. Path requests arrive between each source-destination point (which is chosen at random) according to a Poisson process with an average rate λ, and the holding times are exponentially distributed with mean μ and $\frac{\lambda}{\mu} = 1800$. The bandwidth of input demands are taken to be uniformly distributed between 1 to 3 units. We generate 7000 requests and measure the rejections or blocking for each one of the algorithms. Fig. 5.3 shows the number of the path requests rejected in 20

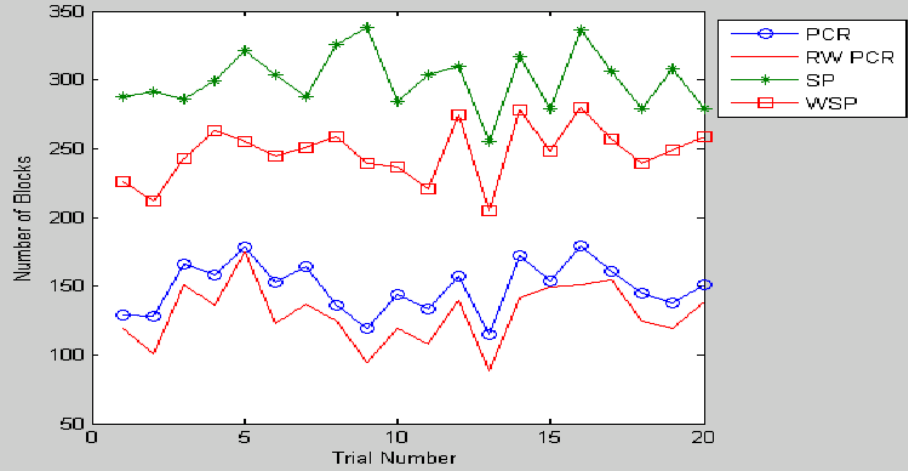


Figure 5.2: Static Case- Result of Applying PCR, RW-PCR, SP, and WSP to the Network under Test

experiments.

From Fig. 5.3, one can see that the number of blocks in RW-PCR is less than other algorithms. PCR has the second best performance and WSP and SP are in next positions. The main reason for the success of RW-PCR is the fact that RW-PCR finds the path that has minimum effect on network criticality (or betweenness sensitivity).

5.5 Behavior of Network Criticality in Sub-Optimal Conditions

In order to investigate the behavior of τ when the optimality condition is not met, we need to explore the properties of the optimization problem introduced in theorem 5.3.3. Our approach is to find an upper bound for the optimality gap (the difference between the optimal and sub-optimal objective values of the optimization problem). The goal is then to minimize this upper bound. In order to establish the upper bound, we use the fact that according to the duality theorem [96], any value of the dual objective function

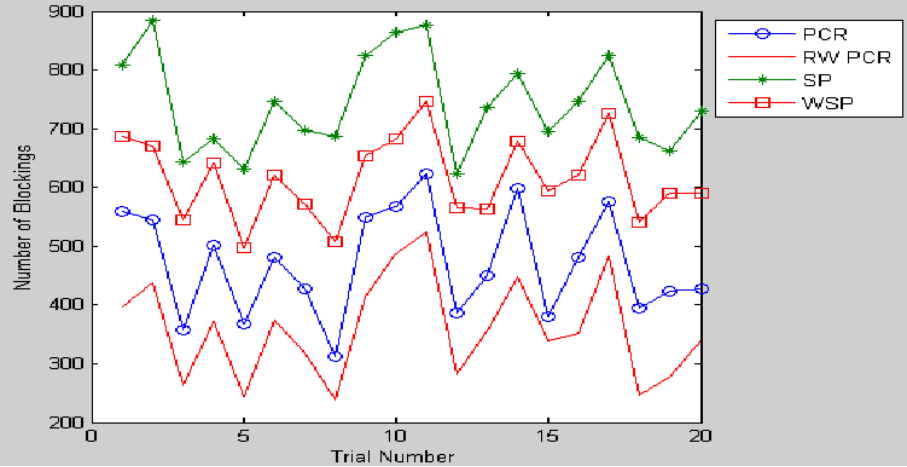


Figure 5.3: Dynamic Case- Result of Applying PCR,RW-PCR, SP, and WSP to the Network under Test

is a lower bound for the optimal value of the primal one. The duality gap, which is the difference between the value of the primal and dual objective functions, provides an upper bound for the optimality gap of a solution for the optimization problem In the optimal case the duality gap is zero because we optimize a strictly convex function of the variables, but in sub-optimal cases, it provides an upper bound for the optimality gap. We try to find this upper bound and use it as a metric to quantify the behavior of network criticality. Note that the optimality gap quantifies the variation of survival value, and it is desired to minimize this variation by applying appropriate control mechanisms. This section tries to provide some directions towards this goal.

5.5.1 Main Result

The main result of this chapter can be summarized in the following theorem.

Theorem 5.5.1 *Consider the following optimization problem:*

$$\begin{aligned} & \textit{Minimize} \quad \tau \\ \textit{Subject to} \quad & \textstyle\sum_{(i,j)\in E} z_{ij} w_{ij} = C \quad , C \textit{ is fixed} \\ & w_{ij} \geq 0 \end{aligned}$$

For any sub-optimal solution of the convex optimization problem, the deviation from optimal solution (optimality gap) has the upper bound of $\frac{\tau}{C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau}{\partial w_{ij}}} (C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau}{\partial w_{ij}} + \tau)$.

∎

Before proving theorem5.5.1, we state one implication of theorem 5.5.1 that leads to an improvement in RW-PCR algorithm.

Observation 5.5.2 *It can be seen from theorem 5.5.1, that the upper bound for* $\frac{\tau - \tau_{dual}}{\tau}$ *is equal to* $1 + \frac{\tau}{C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau}{\partial w_{ij}}}$

Theorem 5.5.1 and observation 5.5.2 can be used as the foundation for building centralized and distributed algorithms for flow assignment and routing problem, as well as network planning. Both centralized and decentralized approaches are possible. Here we show one simple application in traffic engineering. In centralized approach it is enough to consider the statement of observation 5.5.2 as the link weights and find the shortest path(s) using Dijkstra's algorithm:

$$\textit{Dikstra's weight for link } (i,j) = 1 + \frac{\tau}{C \frac{1}{z_{ij}} \frac{\partial \tau}{\partial w_{ij}}} \tag{5.23}$$

To compare the performance of RW-PCR and the modified version of it based on equation 5.23 we ran a scenario on the network of Fig. 5.4.

In each experiment we sent 50000 demands each one with a random bandwidth request between 1 to 3. The traffic was Poisson with offered load of $\frac{\lambda}{\mu} = 1200$. In each experiment the number of blocked requests were counted. This experiment repeated 20 times, then we obtained the average of the blockage in each experiment for RW-PCR and modified RW-PCR. Figure 5.5 shows the results.

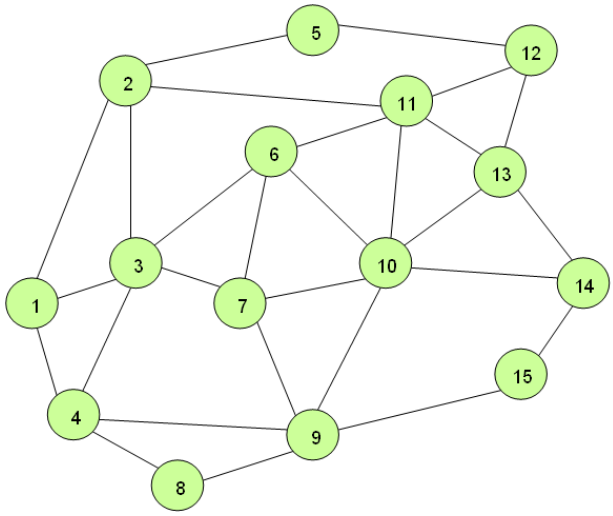
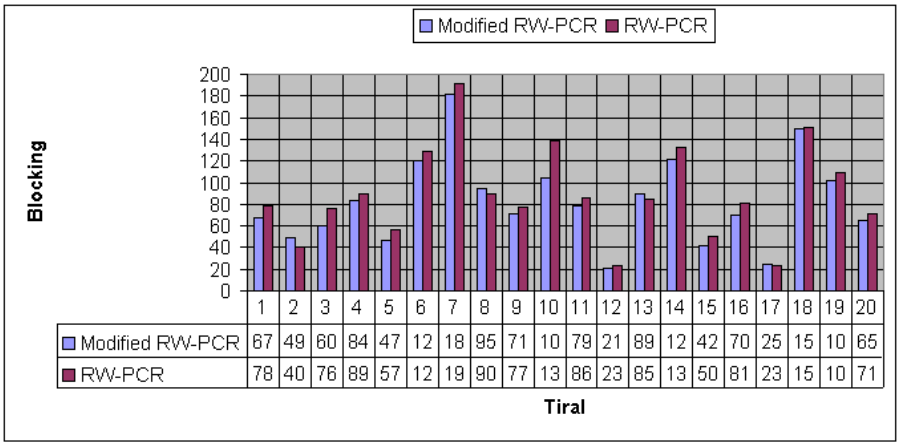


Figure 5.4: Test Network 15 Nodes, 28 Links



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Modified RW-PCR	67	49	60	84	47	12	18	95	71	10	79	21	89	12	42	70	25	15	10	65
RW-PCR	78	40	76	89	57	12	19	90	77	13	86	23	85	13	50	81	23	15	10	71

Figure 5.5: Comparison of RW-PCR and Modified RW-PCR

While equation 5.12 can be used to get the derivative of τ with respect to w_{ij}, we used an approximate approach which is also appropriate when a decentralized algorithm needs to be found. In each iteration we divided the difference of new and old τ to the difference of new weight and old weight. In other words:

$$\frac{\partial \tau}{\partial w_{ij}}|_{t=n+1} \approx \frac{\tau(n+1) - \tau(n)}{w_{ij}(n+1) - w_{ij}(n)} \tag{5.24}$$

where $t = n + 1$ shows the step $n + 1$.

The performance of modified RW-PCR is better than RW-PCR. According to the table of Fig. 5.5, the average of the number of blockages in all the experiments for RW-PCR was 88.9 and it was 82.15 for the modified version.

5.5.2 Dual of τ Optimization Problem

In this section we formulate the dual of the optimization problem 5.17. This is the first step to prove theorem 5.5.1. Using lemma 4.4.1, the optimization problem 5.17 can be written as:

$$\begin{aligned} &Minimize \quad 2nTr(L^+) \\ Subject\ to \quad &\textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} = C \quad , C\ is\ fixed \\ &w_{ij} \geq 0 \end{aligned}$$

Considering equation 5.3 and assuming $\Gamma = L + \frac{J}{n}$, the optimization problem becomes:

$$\begin{aligned} &Minimize \quad 2nTr(\Gamma^{-1}) - 2n \\ Subject\ to \quad &\Gamma = \textstyle\sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t + \frac{J}{n} \\ &\textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} = C \quad , C\ is\ fixed \\ &w_{ij} \geq 0 \end{aligned} \tag{5.25}$$

where we have used an alternative expression for graph Laplacian matrix, that is, $\sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t$ (see appendix A, equation A.1). We write the Lagrangian of the opti-

mization problem 5.25. To this end we first obtain a simple formula for Lagrangian of matrix equation $\Gamma = L + \frac{J}{n}$.

Lemma 5.5.3 *Lagrangian of matrix constraint $\Gamma = L + \frac{J}{n}$ is as follows:*

$$Ł = Tr(T\Gamma) - \sum_{(i,j)\in E} w_{ij} u_{ij}^t T u_{ij} - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1}$$

where T is the $n \times n$ matrix of dual variables.

Proof Let $T = [t_{ij}]$, $\Gamma = [\gamma_{ij}]$, $L = [l_{ij}]$.

$$\begin{aligned}
\Gamma &= L + \frac{J}{n} \\
or\ \gamma_{ij} &= l_{ij} + \frac{1}{n}\ \forall i, j \in N \\
t_{ji}\gamma_{ij} &= t_{ji} l_{ij} + \frac{1}{n} t_{ji}\ \forall i, j \in N \\
\sum_{i,j\in N} t_{ji}\gamma_{ij} &= \sum_{i,j\in N} t_{ji} l_{ij} + \sum_{i,j\in N} \frac{1}{n} t_{ji}\ \forall i, j \in N \\
Tr(T\Gamma) &= Tr(TL) + \sum_{i,j\in N} \frac{1}{n} t_{ji} \qquad (5.26)
\end{aligned}$$

Now if we consider equation $L = \sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t$, we can simplify equation 5.26 as follows:

$$\begin{aligned}
Tr(TL) &= Tr(\sum_{(i,j)\in E} w_{ij} T u_{ij} u_{ij}^t) \\
&= \sum_{(i,j)\in E} w_{ij} Tr(T u_{ij} u_{ij}^t) \\
&= \sum_{(i,j)\in E} w_{ij} u_{ij}^t T u_{ij}
\end{aligned}$$

Here we used the fact that $Tr(T u_{ij} u_{ij}^t) = u_{ij}^t T u_{ij}$. Then:

$$Tr(T\Gamma) = \sum_{(i,j)\in E} w_{ij} u_{ij}^t T u_{ij} + \sum_{i,j\in N} \frac{1}{n} t_{ij} \qquad (5.27)$$

Equation 5.27 shows that the Lagrangian of matrix equation $\Gamma = L + \frac{J}{n}$ can be written as $Tr(T\Gamma) - \sum_{(i,j)\in E} w_{ij} u_{ij}^t T u_{ij} - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1}$. This completes the proof of lemma 5.5.3.

∎

We use lemma 5.5.3 to derive the dual of optimization problem 5.25.

Theorem 5.5.4 *Consider the following convex optimization problem:*

$$\begin{aligned} &Minimize \quad 2nTr(\Gamma^{-1}) - 2n \\ &Subject\ to \quad \Gamma = \textstyle\sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t + \frac{I}{n} \\ &\textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} = C \quad , C\ is\ fixed \\ &w_{ij} \geq 0 \end{aligned}$$

The dual of the optimization problem is as follows:

$$\begin{aligned} maximize \quad & 2Tr(2nT)^{\frac{1}{2}} - 2n - C\lambda - \tfrac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} \\ subject\ to \quad & \tfrac{1}{z_{ij}} u_{ij}^t T u_{ij} \leq \lambda \\ & T\overrightarrow{1} = 2n\overrightarrow{1} \\ & T \geq 0 \end{aligned}$$

where $T \geq 0$ means that T is a positive semi-definite matrix. T and λ are the dual variables.

Proof Using lemma 5.5.3, the Lagrangian of optimization problem 5.25 is:

$$\begin{aligned} L(\Gamma, W, T, \lambda, \rho) &= 2nTr(\Gamma^{-1}) + Tr(T\Gamma) - 2n - C\lambda \\ &+ \sum_{(i,j)\in E} w_{ij}(-u_{ij}^t T u_{ij} + \lambda z_{ij} - \rho_{ij}) - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} \end{aligned}$$

To find the dual formulation, it is enough to take the infimum of the Lagrangian over Γ, W. Considering the fact that $\sum_{(i,j)\in E} t_{ij}\gamma_{ij} = Tr(T\Gamma)$, we will have:

$$\begin{aligned} d(T, \lambda, \rho) &= inf_{\Gamma, W} L(\Gamma, W, T, \lambda, \rho) \qquad (5.28) \\ &= inf_{\Gamma} Tr(2n\Gamma^{-1} + T\Gamma) + inf_{W}(-2n \\ &- \sum_{(i,j)\in E} w_{ij}(-u_{ij}^t T u_{ij} + \lambda z_{ij} - \rho_{ij}) - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} - C\lambda) \end{aligned}$$

The second term in equation 5.28 is minimized if its derivative with respect to all link weights is zero. The minimum of the first term is $-\infty$ unless matrix T is positive

semi-definite, because if T is not positive semi-definite, then its spectrum consists of at least one negative eigenvalue. This implies that if T is not positive semi-definite, then $d(T, \lambda, \rho)$ goes to $-\infty$. Therefore

$$d(T,\lambda,\rho) = \begin{cases} inf_{\Gamma} Tr(2n\Gamma^{-1} + T\Gamma) - \frac{1}{n}\overrightarrow{1}^{t} T \overrightarrow{1} - 2n - C\lambda \\ if - u_{ij}^{t} T u_{ij} + \lambda z_{ij} - \rho_{ij} = 0, \;\; \rho_{ij} \geq 0 \;\; \forall (i,j) \in E \;\; and \;\; T \geq 0 \\ -\infty \;\; otherwise \end{cases} \tag{5.29}$$

where $T = [t_{ij}]$ and $\Gamma = [\gamma_{ij}]$, and $T \geq 0$ means that matrix T is positive semi-definite.

Term $inf_{\Gamma} Tr(2n\Gamma^{-1} + T\Gamma)$ can also be obtained analytically using some known facts from matrix algebra.

Proposition 5.5.5 *For any non-singular square $(n \times n)$ matrix X and any $n \times n$ matrices A and B, we have :*

$$\begin{aligned} \frac{d}{dX} Tr(AX) &= A \\ \frac{d}{dX} Tr(AX^{-1}B) &= -X^{-1}BAX^{-1} \end{aligned}$$

where in general the derivative $\frac{d}{dX} f(X)$ of a scalar-valued differentiable function $f(X)$ of a matrix argument $X \in R^{p \times q}$ is the $q \times p$ matrix whose $(i,j)^{th}$ entry is $\frac{\partial f(X)}{\partial X(j,i)}$ [93].

Proof See [93].

∎

Using proposition 5.5.5 one can find $inf_{\Gamma}(2nTr\Gamma^{-1} + T\Gamma)$ as follows.

$$\begin{aligned} \frac{d}{d\Gamma} Tr(2nTr\Gamma^{-1} + T\Gamma) &= \frac{d}{d\Gamma} Tr(2nTr\Gamma^{-1}) + \frac{d}{d\Gamma} Tr(T\Gamma) = 0 \\ -2n\Gamma^{-1} I \Gamma^{-1} + T &= 0 \\ -2n\Gamma^{-2} + T &= 0 \\ T &= 2n\Gamma^{-2} \qquad (5.30) \\ \Gamma &= (\frac{T}{2n})^{-\frac{1}{2}} \qquad (5.31) \end{aligned}$$

Thus, we have:

$$\begin{aligned} inf_{\Gamma}(2nTr\Gamma^{-1} + T\Gamma) &= Tr(2n(\frac{T}{2n})^{\frac{1}{2}} + T(\frac{T}{2n})^{\frac{-1}{2}}) \\ &= 2Tr(2nT)^{\frac{1}{2}} \end{aligned} \tag{5.32}$$

From equation 5.30 one can see that if L is the optimal solution for Laplacian of the graph, then:

$$T = 2n(L + \frac{J}{n})^{-2} \tag{5.33}$$

Note that

$$T\overrightarrow{1} = 2n(L + \frac{J}{n})^{-2}\overrightarrow{1}$$

But we know:

$$\begin{aligned} (L + \frac{J}{n})\overrightarrow{1} &= L\overrightarrow{1} + \frac{1}{n}J\overrightarrow{1} \\ &= 0 + \frac{1}{n} \times n\overrightarrow{1} = \overrightarrow{1} \end{aligned}$$

Therefore:

$$\begin{aligned} (L + \frac{J}{n})^{-1}(L + \frac{J}{n})\overrightarrow{1} &= (L + \frac{J}{n})^{-1}\overrightarrow{1} \\ (L + \frac{J}{n})^{-1}\overrightarrow{1} &= (L + \frac{J}{n})^{-2}\overrightarrow{1} \\ (L + \frac{J}{n})^{-2}\overrightarrow{1} &= \overrightarrow{1} \\ 2n(L + \frac{J}{n})^{-2}\overrightarrow{1} &= 2n\overrightarrow{1} \end{aligned}$$

So:

$$T\overrightarrow{1} = 2n\overrightarrow{1} \tag{5.34}$$

Equation 5.34 is valid for optimal solution of the optimization problem 5.28. Hence, we can use it as a constraint in our optimization problem.

We need one more step to take before writing the dual optimization problem. We can remove the dual variable set ρ_{ij} considering the fact that $\rho_{ij} \geq 0$. Using this inequality in the constraint part of equation 5.29 results in:

$$\begin{aligned} -u_{ij}^t T u_{ij} + \lambda z_{ij} - \rho_{ij} &= 0 \ \ and \ \ \rho_{ij} \geq 0 \\ -u_{ij}^t T u_{ij} + \lambda z_{ij} &\geq 0 \\ \frac{1}{z_{ij}} u_{ij}^t T u_{ij} &\leq \lambda \end{aligned}$$

Hence, the dual optimization problem can be written as follows:

$$\begin{aligned} &maximize && 2Tr(2nT)^{\frac{1}{2}} - 2n - C\lambda - \tfrac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} && (5.35) \\ &subject\ \ to && \tfrac{1}{z_{ij}} u_{ij}^t T u_{ij} \leq \lambda && (5.36) \\ & && T\overrightarrow{1} = 2n\overrightarrow{1} && (5.37) \\ & && T \geq 0 \end{aligned}$$

This completes the proof of theorem 5.5.4.

∎

5.5.3 Orthogonalization

We observe that according to equation 5.30 for the optimal solution of the optimization problem, one can write

$$\begin{aligned} T &= 2n\Gamma^{-2} \\ &= 2n(L + \frac{J}{n})^{-2} \end{aligned}$$

Using equation 5.3 we have

$$\begin{aligned} T &= 2n(L^+ + \frac{J}{n})^2 \qquad (5.38) \\ &= 2n(L^+)^2 + 2n \times \frac{1}{n^2} \times n \times J \end{aligned}$$

We notice that $L^+J = 0$ and $J \times J = nJ$, therefore

$$T = 2n(L^+)^2 + 2J \tag{5.39}$$

Equation 5.39 shows that matrix $T-2J$ is proportional to the square of L^+. Motivated by this, we define a new variable $Q = (\frac{T-2J}{\lambda})^{\frac{1}{2}}$. We use λ in the denominator of Q to remove it from the constraint 5.36.

Now we use equation 5.39 to obtain matrix Q for the optimal case.

$$\begin{aligned} Q &= (\frac{T-2J}{\lambda})^{\frac{1}{2}} \\ &= (\frac{2nL^{+2}+2J-2J}{\lambda})^{\frac{1}{2}} \\ &= (\frac{2n}{\lambda})^{\frac{1}{2}}L^+ \end{aligned} \tag{5.40}$$

According to equation 5.40, projection of Q on vector $\overrightarrow{1}$ is zero (because $L^+\overrightarrow{1} = 0$).

$$\begin{aligned} Q\overrightarrow{1} &= (\frac{2n}{\lambda})^{\frac{1}{2}}L^+\overrightarrow{1} \\ Q\overrightarrow{1} &= 0 \end{aligned} \tag{5.41}$$

Equation 5.41 shows that the span of matrix Q is in the space orthogonal to vector $\overrightarrow{1}$ (similar to Laplacian matrix), in contrast to the span of matrix T which includes vector $\overrightarrow{1}$.

We apply this change of variable in optimization problem 5.35.

Lemma 5.5.6 *Let $Q = (\frac{T-2J}{\lambda})^{\frac{1}{2}}$. If we apply this change of variable in optimization problem 5.35, the new optimization problem will be:*

$$\begin{aligned} \textit{maximize} \quad & 2nC^{-1}(Tr(Q))^2 \\ \textit{subject to} \quad & \tfrac{1}{\sqrt{z_{ij}}}\left\|Q_i - Q_j\right\| \leq 1 \\ & Q\overrightarrow{1} = 0 \\ & Q \geq 0 \end{aligned}$$

Proof Applying the variable change to the dual function results in:

$$\begin{aligned}(2nT)^{\frac{1}{2}} &= (2n)^{\frac{1}{2}}(\lambda Q^2 + 2J)^{\frac{1}{2}} \\ &= (2n)^{\frac{1}{2}}(\lambda Q^2 + (\frac{2}{n})J^2)^{\frac{1}{2}} \\ &= (2n)^{\frac{1}{2}}(\lambda^{\frac{1}{2}}Q + (\frac{2}{n})^{\frac{1}{2}}J) \\ &= (2n\lambda)^{\frac{1}{2}}Q + 2J \end{aligned} \tag{5.42}$$

Therefore:

$$\begin{aligned}Tr((2nT)^{\frac{1}{2}}) &= (2n\lambda)^{\frac{1}{2}}Tr(Q) + 2Tr(J) \\ &= (2n\lambda)^{\frac{1}{2}}Tr(Q) + 2n \end{aligned} \tag{5.43}$$

Now we apply equation 5.43 to the dual function 5.35.

$$\begin{aligned}d(T,\lambda) &= 2Tr(2nT)^{\frac{1}{2}} - 2n - C\lambda - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} \\ &= 2(2n\lambda)^{\frac{1}{2}}Tr(Q) + 4n - 2n - C\lambda - \frac{1}{n}\lambda \overrightarrow{1}^t Q^2 \overrightarrow{1} - \frac{1}{n} \times 2\overrightarrow{1}^t \overrightarrow{1}\overrightarrow{1}^t \overrightarrow{1} \\ &= 2(2n\lambda)^{\frac{1}{2}}Tr(Q) + 4n - 2n - C\lambda - 0 - 2n \\ &= 2(2n\lambda)^{\frac{1}{2}}Tr(Q) - C\lambda \end{aligned} \tag{5.44}$$

Note that Q is symmetric (because L^+ is symmetric) and $Ju_{ij} = 0$, therefore the first constraint of the dual problem 5.35 (constraint 5.36) can be written in terms of new variable Q as follows:

$$\begin{aligned}\frac{1}{z_{ij}}u_{ij}^t T u_{ij} &\leq \lambda \\ \frac{1}{z_{ij}}(\lambda u_{ij}^t Q^2 u_{ij} + u_{ij}^t J u_{ij}) &\leq \lambda \\ \frac{1}{z_{ij}}(\lambda (u_{ij}^t Q^t)(Q u_{ij}) + 0) &\leq \lambda \\ \frac{1}{z_{ij}}\lambda (Q u_{ij})^t (Q u_{ij}) &\leq \lambda \\ \frac{1}{z_{ij}}\left\|Q u_{ij}\right\|^2 &\leq 1 \\ \frac{1}{\sqrt{z_{ij}}}\left\|Q_i - Q_j\right\| &\leq 1 \end{aligned}$$

where Q_i denotes the i^{th} column of matrix Q. Our dual optimization problem now can be written as:

$$\begin{aligned} &maximize && 2(2n\lambda)^{\frac{1}{2}}Tr(Q) - C\lambda && (5.45)\\ &subject\ \ to && \tfrac{1}{\sqrt{z_{ij}}}\left\|Q_i - Q_j\right\| \le 1 && (5.46)\\ & && Q\overrightarrow{1} = 0 &\\ & && Q \ge 0 & \end{aligned}$$

The maximization over λ in optimization problem 5.45 can be done analytically.

$$\begin{aligned} d(Q,\lambda) &= 2(2n\lambda)^{\frac{1}{2}}Tr(Q) - C\lambda \\ \frac{\partial d}{\partial \lambda} = 0 &\Rightarrow 2 \times (2n)^{\frac{1}{2}} \times \frac{1}{2}\lambda^{-\frac{1}{2}}Tr(Q) - C = 0 \\ \lambda &= 2nC^{-2}(Tr(Q))^2 && (5.47)\\ d(Q) &= 2(2n)^{\frac{1}{2}} \times (2n)^{\frac{1}{2}}C^{-1}(Tr(Q))^2 - C \times 2nC^{-2}(Tr(Q))^2 \\ &= 2nC^{-1}(Tr(Q))^2 && (5.48) \end{aligned}$$

Using equation 5.48, the final form of our dual optimization problem would be

$$\begin{aligned} &maximize && 2nC^{-1}(Tr(Q))^2 && (5.49)\\ &subject\ \ to && \tfrac{1}{\sqrt{z_{ij}}}\left\|Q_i - Q_j\right\| \le 1 && (5.50)\\ & && Q\overrightarrow{1} = 0 &\\ & && Q \ge 0 & \end{aligned}$$

This completes the proof of lemma 5.5.6.

∎

We can now find an expression for matrix Q.

Lemma 5.5.7 *Matrix Q in optimization problem 5.49 is equal to:*

$$Q = \frac{1}{\max_{(i,j)\in E} \frac{1}{\sqrt{z_{ij}}}\left\|L_i^+ - L_j^+\right\|}L^+$$

Proof Since $\frac{1}{z_{ij}} u_{ij} T u_{ij}^t \leq \lambda$ (inequality 5.36), we can find the value of λ for optimal solution as follows:

$$\begin{aligned}
\lambda &= \max \frac{1}{z_{ij}} u_{ij} T u_{ij}^t \\
&= \max \frac{1}{z_{ij}} u_{ij} 2n\Gamma^{-2} u_{ij}^t \\
&= \max \frac{2n}{z_{ij}} u_{ij} 2nL^{+2} u_{ij}^t \\
&= \max \frac{2n}{z_{ij}} \left\| L^+ u_{ij} \right\|^2 \qquad (5.51) \\
&= \max \frac{2n}{z_{ij}} \left\| L_i^+ - L_j^+ \right\|^2 \qquad (5.52)
\end{aligned}$$

Replacing equation 5.52 in 5.40 we get

$$\begin{aligned}
Q &= (\frac{2n}{\lambda})^{\frac{1}{2}} L^+ \\
&= (2n)^{\frac{1}{2}} \frac{1}{(2n \max_{(i,j)\in E} \frac{1}{z_{ij}} \left\| L_i^+ - L_j^+ \right\|^2)^{\frac{1}{2}}} L^+ \\
&= \frac{1}{\max_{(i,j)\in E} \frac{1}{\sqrt{z_{ij}}} \left\| L_i^+ - L_j^+ \right\|} L^+ \qquad (5.53)
\end{aligned}$$

This completes the proof of lemma 5.5.7.

∎

Equation 5.53 gives matrix Q based on the Moore-Penrose inverse of the graph Laplacian L^+.

5.5.3.1 Proof of Theorem 5.5.1

We have now enough information to prove one of our main result, theorem 5.5.1.

Proof The value of the objective function of dual problem 5.49 can be obtained by applying equation 5.53 to equation 5.49.

$$\begin{aligned}
\tau_{dual} &= 2nC^{-1} Tr(Q)^2 |_{Q = \frac{1}{\max_{(i,j)\in E} \frac{1}{\sqrt{z_{ij}}} \left\| L_i^+ - L_j^+ \right\|} L^+} \\
&= 2nC^{-1} \frac{1}{\max_{(i,j)\in E} \frac{1}{z_{ij}} \left\| L_i^+ - L_j^+ \right\|^2} (TrL^+)^2 \qquad (5.54)
\end{aligned}$$

Let W^* be the optimal weight matrix for the optimization problem, and let τ^* denote the optimal value of τ, that is:

$$\tau^* = \tau|_{W=W^*}$$

According to the duality theorem, the objective value of the dual problem is a lower bound for the optimal objective value of the primal optimization problem, thus:

$$\begin{aligned} \tau^* &\geq \tau_{dual} \\ \tau - \tau^* &\leq \tau - \tau_{dual} \end{aligned} \tag{5.55}$$

Inequality 5.55 shows that the duality gap $\tau - \tau_{dual}$ provides an upper bound for the optimality gap $\tau - \tau^*$. On the other hand, the duality gap can be obtained as follows:

$$\begin{aligned} \tau - \tau_{dual} &= 2nTrL^+ - 2nC^{-1}\frac{1}{\max_{(i,j)\in E}\frac{1}{z_{ij}}\left\|L_i^+ - L_j^+\right\|^2}(TrL^+)^2 \\ &= 2nTrL^+ + (2n)^2C^{-1}\frac{1}{-2n\max_{(i,j)\in E}\frac{1}{z_{ij}}\left\|L_i^+ - L_j^+\right\|^2}(TrL^+)^2 \\ &= \tau + C^{-1}\frac{\tau^2}{\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}}} \\ &= \frac{\tau}{C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}}}(C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}} + \tau) \end{aligned} \tag{5.56}$$

This completes the proof of theorem 5.5.1.

∎

In case that we don't have the optimal solution, we should have $C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}} + \tau < 0$ which is clearly a result of equation 5.56. When we do not have the optimal solution:

$$\begin{aligned} \tau - \tau_{dual} &> 0 \\ \frac{\tau}{C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}}}(C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}} + \tau) &> 0 \\ C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial\tau}{\partial w_{ij}} + \tau &< 0 \end{aligned} \tag{5.57}$$

Here we used the fact that $\frac{\partial\tau}{\partial w_{ij}} < 0$.

Observation 5.5.8 *All the results that we have obtained for network criticality (τ) are applicable for normalized network criticality ($\hat{\tau}$) as well, with just a normalization factor $\frac{1}{n(n-1)}$ since $\hat{\tau} = \frac{1}{n(n-1)}\tau$.*

5.6 Network Design Problem

One of the important applications of our survival value analysis is to design robust networks. Network criticality is again the main reference. Lets take the case of core networks. Reliable data transport is the main service of a core network. Backbone networks may be congested in different times of the day regularly (based of a predicted traffic pattern) or irregularly caused by unpredicted disturbances.

In designing core networks we need to consider the robustness of the network to these unwanted changes. In other words we need to design a network with reasonable average network criticality. The same is true when we re-plan a network. For instance in AutoNet the general topology manager may not be able to accommodate customer demands with the present network parameters, then a process is activated in long-term loop to re-plan the network accordingly. The main goal in this re-dimensioning phase is also to keep τ as low as possible.

In the previous sections of this chapter we provided the details of the optimization problem to minimize τ. We can use those results to derive appropriate algorithms to minimize the duality gap of the network criticality for different purposes (modified RW-PCR is one instance of such algorithms which is used for traffic management). In the following we propose an alternate approach to the weight assignment problem, which is useful when we want to use an offline method to find the exact solution of the optimization problem. This approach is based on the idea of semi-definite programming. The method is very simple but effective.

5.6.1 Semi-Definite Programming Approach

Network criticality is a nonlinear function of weights. There are some powerful software packages to solve nonlinear optimization problems (see [98] as an elegant example), however, all of which have some limitations and cannot provide solution for some nonlinear objectives. In some of the nonlinear optimization problems, it is possible to convert the optimization problem to a *Semi-definite Programming* [99, 100, 101]. Semi-definite programming can be viewed as an extension for linear programming where the objective and constraints are linear combinations of semi-definite matrices (see appendix B for a brief review of semi-definite programming concepts). Semi-definite programs can be efficiently solved in a reasonable time.

Optimization problem 5.17 can be converted to a semi-definite programming problem. Suppose $\Gamma = (L + \frac{J}{n})^{-1}$. In order to have a semi-definite program we need to have the constraints of the optimization as linear functions of semi-definite matrices. In fact Γ can be written as a semi-definite inequality. We consider matrix $\Theta = \begin{pmatrix} \Gamma & I \\ I & L + \frac{J}{n} \end{pmatrix}$. The necessary and sufficient condition for positive semi-definiteness of Θ is that its Schur complement (see appendix B) be positive semi-definite. The Schur complement of Θ is $\Gamma - (L + \frac{J}{n})^{-1}$.

$$\Theta = \begin{pmatrix} \Gamma & I \\ I & L + \frac{J}{n} \end{pmatrix} \geq 0 \Leftrightarrow \Gamma \geq (L + \frac{J}{n})^{-1} \tag{5.58}$$

where $\geq$ means positive semi-definite. Since the optimization problem 5.17 should minimize $Tr(\Gamma)$, the equality in equation 5.58 is chosen which is equal to $\Gamma = (L + \frac{J}{n})^{-1}$. The Constraint of optimization problem 4.12 can also be represented in semi-definite form as follows:

$$\begin{aligned} \sum_{(i,j)inE} z_{ij} w_{ij} &= C \\ Tr(Diag(Vec(Z)) \times Diag(Vec(W))) &= C \end{aligned}$$

Note that $Diag(Vec(W))$ is a diagonal matrix with w_{ij}'s in main diagonal. This matrix is positive semi-definite because $w_{ij} \geq 0 \quad \forall (i,j) \in E$. Now optimization problem 5.17 can be converted to a semi-definite programming. We write the semi-definite optimization problem for normalized criticality, $\hat{\tau}$, as in practical cases $\hat{\tau}$ is more useful, particularly when we need to compare the robustness of different networks. According to observation 5.5.8, the optimization problem is the same as problem 5.17 except the objective which is divided by $\frac{1}{n(n-1)}$.

$$\begin{gathered} Minimize \quad \tfrac{2}{n-1} Tr(\Gamma) \\ Subject\ to \quad Tr(Diag(Vec(Z)) \times Diag(Vec(W))) = C \\ \begin{pmatrix} \Gamma & I \\ I & L + \frac{J}{n} \end{pmatrix} \geq 0 \\ Diag(Vec(W)) \geq 0 \end{gathered} \tag{5.59}$$

This new optimization problem can be solved with standard methods of solving semi-definite programs. Later in this chapter we solve optimization problem 5.59 for some specific graphs to show how the concept of network criticality helps find robust network topologies.

5.6.2 Capacity Planning

In this section we consider the capacity planning as an important special case of network design problem. Consider a network $G(N, E, W)$ where the link weights are equal to the link capacities, that is, $w_{ij} = c_{ij} \ \forall (i,j) \in E$ (c_{ij} denotes the capacity of link (i,j)). We investigate the capacity assignment problem in which network topology and traffic load $\gamma_{ij} \ \forall (i,j) \in E$ are assumed known and fixed. The goal is to find the capacity of the links so as to minimize the network criticality under the constraint that the total cost of the planning is fixed. Let z_{ij} be the symmetric cost of assigning capacity c_{ij} to link (i,j), and suppose that we have a liner cost function. The total cost

of the capacity assignment problem is $\sum_{(i,j)\in E} z_{ij}c_{ij}$. We fix this total cost to C. We can write the optimization problem for capacity assignment as follows:

$$\begin{aligned} & Minimize \quad \tau \\ Subject\ to \quad & \textstyle\sum_{(i,j)\in E} c_{ij}z_{ij} = C \quad , C\ is\ fixed \\ & c_{ij} \geq \gamma_{ij} \end{aligned} \tag{5.60}$$

By applying the change of variable $c_{ij} = c'_{ij} + \gamma_{ij}$ to the optimization problem 5.60, we will have the following convex optimization problem.

$$\begin{aligned} & Minimize \quad \tau \\ Subject\ to \quad & \textstyle\sum_{(i,j)\in E} c'_{ij}z_{ij} = C' \quad , C'\ is\ fixed \\ & c'_{ij} \geq 0 \end{aligned} \tag{5.61}$$

where $C' = C - \sum_{(i,j)\in E} z_{ij}\gamma_{ij}$. The optimization problem 5.61 is now converted to the optimization problem 5.17 (with $w_{ij} \rightarrow c'_{ij}$ and $C \rightarrow C'$), therefore, all the results of this chapter are applicable for the capacity assignment problem. Later in this chapter we will see when the optimization problem 5.61 is equal to the Kleinrock's capacity assignment problem [87].

5.7 Case study

In this section we will find the optimal weight matrix for some well-structured graphs where the analytical study is possible. We also give some examples of network design using semi-definite programming approach.

5.7.1 Complete Graph on n Nodes (K_n)

For K_n we can obtain the solution of optimization problem 5.17 analytically. In this example we assume $z_{ij} = 1 \ \ \forall\ (i,j) \in E$. In order to find the optimal weight set for K_n we need the following lemma.

Lemma 5.7.1 *Consider optimization problem 5.59 and suppose all the links have equal costs. If there is an automorphism on a graph G(N,E,W) that can map link $l = (i, j)$ on link $l' = (i', j')$, then these links should have equal optimal weights.*

Proof Let G′ be the new graph after applying the automorphism. Any automorphism on graph G can be shown by a matrix T so that the Laplacian of transformed graph G′ can be obtained from Laplacian of original graph G as $L(G') = TL(G)T^t$. This means that L(G) and L(G′) have the same eigenvalues. As a result according to the Lemma 4.4.2 criticality of graph G and G′ are the same: $\hat{\tau}(G) = \hat{\tau}(G')$. On the other hand the solution of optimization problem 5.17 is unique. As a result the weight of link l and l′ are the same.

∎

Corollary 5.7.2 *Consider optimization problem 5.59 and assume that the graph of the network is an edge-transitive graph with equal link costs. The optimal weight for a link $(i, j) \in E$ is equal to $w_{ij} = \frac{C}{m}$, where m denotes the number of graph links.*

Proof A graph is edge-transitive, if there is an automorphism that can map any two links of the graph. According to lemma 5.7.1 all the link weights are equal. In addition, suppose $w_{ij} = w \ \forall \ (i, j) \in E$, then constraint $\sum_{(i,j)\in E} w_{ij} = C$ implies that $w = \frac{C}{m}$. This completes the proof of corollary 5.7.2.

∎

Complete graph K_n is an edge-transitive graph, therefore, according to corollary 5.7.2 the optimal weight of all the links of K_n are equal. Let denote this common weight by w. Further, let vector X be the eigenvector of Laplacian matrix for eigenvalue λ. Then:

$$\begin{aligned} L(K_n) &= w(nI - J) \\ LX &= \lambda X \\ w((n-1)x_j - \sum_{i \neq j} x_i) &= \lambda x_j \quad for \ \ i \neq 1 \end{aligned}$$

In addition $\sum_{i=1}^{n} x_i = 0$ (see appendix A). Therefore

$$\begin{aligned} w((n-1)x_j - (-x_j)) &= \lambda x_j \\ \lambda_i &= nw \ \ for \ \ i = 2 \ldots n \end{aligned}$$

One can also find link weight w from corollary 5.7.2. The total number of links in K_n is $m = n(n-1)$, therefore, according to the corollary 5.7.2:

$$w = \tfrac{1}{n(n-1)} C$$

The eigenvalues of K_n are:

$$\lambda_i = \tfrac{1}{n-1} C \quad \forall \, i \in N \tag{5.62}$$

It is easy now to calculate network criticality for graph K_n using Lemma 4.4.2 and equation 5.62.

$$\begin{aligned} \hat{\tau} &= \frac{2}{n-1} \sum_{i=2}^{n} \frac{1}{\lambda_i} \\ &= \frac{2}{n-1}(n-1)(n-1)\frac{1}{C} \\ &= \frac{2(n-1)}{C} \end{aligned} \tag{5.63}$$

According to equation 5.63 the optimal weight of a link in a complete graph is linearly increasing with the size of the network. This provides a basis for comparing the normalized network criticality of different networks against the full-mesh on n nodes.

5.7.1.1 Case of Unequal Link Costs for K_n

When the link costs (z_{ij}'s) are not equal, we can use the semi-definite approach to find the best weight assignment. We use a numerical example to show the effect of changes in link costs. We consider the complete graph on 6 nodes (K_6), and we assume that the

matrix of link costs is given as follows:

$$Z = [z_{ij}] = \begin{pmatrix} 0 & 1.2 & 1 & 1 & 1 & 2 \\ 1.2 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 2 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Further, let $C = 2000$. We used the semi-definite form of the optimization problem which is described in equation 5.59 and solved semi-definite program for complete graph K_6 using `CVX`, a package for specifying and solving convex programs [102], [103]. The optimal weight assignment is given in the following.

$$W = [w_{ij}] = \begin{pmatrix} 0 & 54.9820 & 85.7652 & 85.7463 & 85.7378 & \mathbf{0.0031} \\ 54.9820 & 0 & 64.0572 & 63.9948 & 63.9753 & 78.0271 \\ 85.7652 & 64.0572 & 0 & 54.3210 & 54.2332 & 81.1880 \\ 85.7463 & 63.9948 & 54.3210 & 0 & 54.4351 & 81.2583 \\ 85.7378 & 63.9753 & 54.2332 & 54.4351 & 0 & 81.2762 \\ \mathbf{0.0031} & 78.0271 & 81.1880 & 81.2583 & 81.2762 & 0 \end{pmatrix}$$

The weight matrix shows that the optimal weight assignment is not uniform. The optimal weight of link $(1,6)$ is $w_{16} = w_{61} = 0.0031$ which means that link $(1,6)$ is effectively down. In other words, the topology of the optimal network is not K_6 any more.

5.7.2 Optimal Network Criticality for a Tree

In order to find the optimal weight set for a tree we use equation 4.53.

$$\begin{aligned} \tau &= \sum_{(i,j)\in E} \frac{n_{ij}}{w_{ij}} \\ \Rightarrow \frac{\partial \tau}{\partial w_{ij}} &= -\frac{n_{ij}}{w_{ij}^2} \end{aligned} \tag{5.64}$$

But from equation 5.20 we know that for optimal weights $C\frac{\partial\tau}{\partial w_{ij}} + \tau z_{ij} = 0$. By using this relation in equation 5.64 we get:

$$\frac{\partial\tau}{\partial w_{ij}} = -\frac{n_{ij}}{w_{ij}^2} = -\frac{z_{ij}\tau}{C} \tag{5.65}$$

$$\Rightarrow w_{ij} = (\frac{n_{ij}C}{z_{ij}\tau})^{\frac{1}{2}} \tag{5.66}$$

From the constraint of the optimization problem we have $\sum_{(i,j)\in E} z_{ij}w_{ij} = C$. Hence:

$$\sum_{(i,j)\in E} (\frac{n_{ij}z_{ij}C}{\tau})^{\frac{1}{2}} = C \tag{5.67}$$

$$\tau = (\sum_{(i,j)\in E} (\frac{n_{ij}z_{ij}}{C})^{\frac{1}{2}})^2 \tag{5.68}$$

Now it is enough to substitute τ from equation 5.68 in equation 5.66 to have optimal weight for tree.

$$w_{ij} = (\frac{n_{ij}C}{z_{ij}})^{\frac{1}{2}} \times \frac{1}{\sum_{(i,j)\in E}(\frac{n_{ij}z_{ij}}{C})^{\frac{1}{2}}}$$

Finally

$$w_{ij} = \frac{C}{z_{ij}} \times \frac{(n_{ij}z_{ij})^{\frac{1}{2}}}{\sum_{(i,j)\in E}(n_{ij}z_{ij})^{\frac{1}{2}}} \tag{5.69}$$

Equation 5.69 shows that the optimal weight of a link in a tree is proportional to the square root of n_{ij}.

5.7.2.1 Capacity Planning for a Tree

The capacity assignment problem for a tree can be solved analytically using the guidelines provided in section 5.6.2. It is enough to apply the following changes in equation 5.69:

$$w_{ij} \rightarrow c_{ij} - \gamma_{ij}$$

$$C \rightarrow C - \sum_{(i,j)\in E} z_{ij}\gamma_{ij}$$

The optimal capacity assignment for a tree would be:

$$c_{ij} = \gamma_{ij} + \frac{C - \sum_{(i,j)\in E} z_{ij}\gamma_{ij}}{z_{ij}} \times \frac{(n_{ij}z_{ij})^{\frac{1}{2}}}{\sum_{(i,j)\in E}(n_{ij}z_{ij})^{\frac{1}{2}}} \tag{5.70}$$

There is a close analogy between our result and Kleinrock's result for capacity assignment. In [87] Kleinrock showed that under the independence assumption the optimal capacity (to minimize average delay of the network) of a link is proportional to the square root of the link rate. Note that for a tree n_{ij} is proportional to the link load (γ_{ij}) since there is only one path between every source-destination pair. As a result, equation 5.70 is similar to the Kleinrock's equation for optimal capacity ([87], §5.7, equation 5.26). This result is not surprising because the network criticality of a tree according to equation 4.53 is equal to $\tau = \sum_{(i,j)\in E} \frac{n_{ij}}{c_{ij}-\gamma_{ij}}$ (considering $w_{ij} = c'_{ij}$). This is the same expression that is used in [87] to find the average delay of a network ([87], §5.6, equation 5.19), therefore, the minimization of network criticality is equal to the minimization of the average network delay when the network is a tree (an acyclic connected graph).

5.7.3 Hypercube with 2^n nodes (H_n)

Now we consider hypercube of order n (H_n), another well-structured graph whose criticality can be obtained analytically. In this example we assume $z_{ij} = 1 \ \ \forall \ (i,j) \in E$. Hypercube is an edge-transitive graph, therefore, by corollary 5.7.2 the optimal solution of the optimization problem 5.59 for hypercube has equal weights. Hence, we consider a hypercube with weight w for all the links. Hypercube can be recursively built by the use of "Cartesian Product" of a graph with K_2 (complete graph on 2 nodes):

$$H_{n+1} = H_n \Box K_2;$$

where $\Box$ denotes the cartesian product. This equation can also be written using "Kronecker Product" (see appendix C for the definition of cartesian and Kronecker

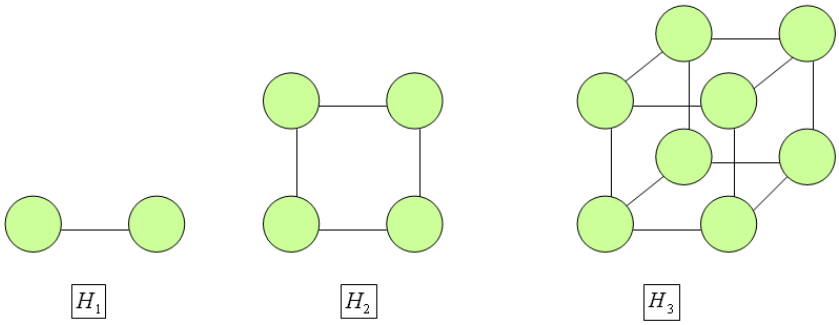


Figure 5.6: Hypercube Topology (H_1, H_2, H_3)

products of two graphs):

$$\begin{aligned} H_{n+1} &= H_n \square K_2; \\ &= H_n \otimes I_2 + I_{2^n} \otimes K_2 \end{aligned} \tag{5.71}$$

We have used the symbol $\otimes$ to denote Kronecker product. Fig. 5.6 shows hypercube topology for $n = 1$ *to* 3.

We try to obtain eigenvalues of adjacency matrix of H_n using equation 5.71. We find the eigenvalues for $w = 1$, then we multiply these normalized eigenvalues by w to find the eigenvalues for the general case.

$$\begin{aligned} H_{n+1} &= H_n \otimes I_2 + I_{2^n} \otimes K_2 \\ &= \begin{pmatrix} H_n & 0 \\ 0 & H_n \end{pmatrix} + \begin{pmatrix} 0 & I_{2^n} \\ I_{2^n} & 0 \end{pmatrix} \\ H_{n+1} &= \begin{pmatrix} H_n & I_{2^n} \\ I_{2^n} & H_n \end{pmatrix} \end{aligned}$$

For simplicity of notation, we drop the subscript from I_{2^n} and use I instead, which means the identity matrix of appropriate order. Now we try to build the determinant

of characteristic matrix $H_{n+1} - \lambda I$:

$$\begin{aligned} H_{n+1} - \lambda I &= \begin{pmatrix} H_n - \lambda I & I \\ I & H_n - \lambda I \end{pmatrix} \\ d_{n+1} &= |H_{n+1} - \lambda I| = \det \begin{pmatrix} H_n - \lambda I & I \\ I & H_n - \lambda I \end{pmatrix} \\ &= \det \begin{pmatrix} I & H_n - \lambda I \\ H_n - \lambda I & I \end{pmatrix} \end{aligned}$$

Now we multiply the first row by $H_n - \lambda I$, and then subtract the first row from the second row. We have:

$$\begin{aligned} d_{n+1}(\lambda) &= \det \begin{pmatrix} I & H_n - \lambda I \\ H_n - \lambda I - (H_n - \lambda I) & I - (H_n - \lambda I)^2 \end{pmatrix} \\ &= \det \begin{pmatrix} I & H_n - \lambda I \\ 0 & I - (H_n - \lambda I)^2 \end{pmatrix} \\ &= |I - (H_n - \lambda I)^2| \\ &= |H_n - (\lambda - 1)I||H_n - (\lambda + 1)I| \\ d_{n+1}(\lambda) &= d_n(\lambda - 1)d_n(\lambda + 1) \end{aligned}$$

Using this recursive formula for determinant of hypercube, one can find with induction that the eigenvalues of H_n are $2k - n,\ \ k = 0, 1, ..., n$ with multiplicity $C(n, k) = \frac{n!}{k!(n-k)!}$. This result is true when all the weights are set to 1. In general case where we have a weight w for each link, the eigenvalue is also multiplied by this weight.

We notice that hypercube is a regular graph (degree of all nodes are n). This means that we can find eigenvalues of Laplacian using eigenvalues of the adjacency matrix of H_n:

$$\begin{aligned} L_n = nI - H_n \Rightarrow \lambda_k &= n - (2k - n) \textit{ with multiplicity } C(n, k) \\ \lambda_k &= 2(n - k) \textit{ with multiplicity } C(n, k) \end{aligned}$$

Now one can easily find the network criticality for H_n using lemma 4.4.2.

$$\begin{aligned}\hat{\tau} &= \frac{2}{2^n-1}\sum_k \frac{1}{\lambda_k} \\ &= \frac{2}{2^n-1}\sum_{k=0}^{n-1}\frac{C(n,k)}{2(n-k)w} \\ &= \frac{1}{(2^n-1)w}\sum_{k=0}^{n-1}\frac{C(n,k)}{n-k}\end{aligned} \tag{5.72}$$

On the other hand form the constraint of the optimization problem 5.17, it is clear that:

$$\begin{aligned}\sum_{(i,j)\in E} w_{ij} &= C \\ n2^n w &= C \\ w &= \frac{C}{n2^n}\end{aligned} \tag{5.73}$$

The final expression for network criticality of H_n can be found by applying equation 5.73 in 5.72:

$$\begin{aligned}\hat{\tau} &= \frac{n2^n}{(2^n-1)C}\sum_{k=0}^{n-1}\frac{C(n,k)}{n-k} \\ \hat{\tau} &= \frac{n}{(1-\frac{1}{2^n})C}\sum_{k=0}^{n-1}\frac{C(n,k)}{n-k} \\ \hat{\tau} &= \frac{n}{(1-\frac{1}{2^n})C}\sum_{i=1}^{n}\frac{C(n,i)}{i}\end{aligned} \tag{5.74}$$

To obtain the last equation we applied the change of variable $i = n - k$ and used the fact that $C(n, n-i) = C(n, i)$. Equation 5.74 shows the behavior of normalized network criticality when the size of hypercube increases. Fig. 5.7 shows the normalized network criticality of hypercube for $n = 1$ to $n = 7$.

We can also compare the normalized criticality of a hypercube H_n with a complete graph K_{2^n} to see how the robustness is decreased by changing a complete graph to a hypercube (with the same number of nodes).

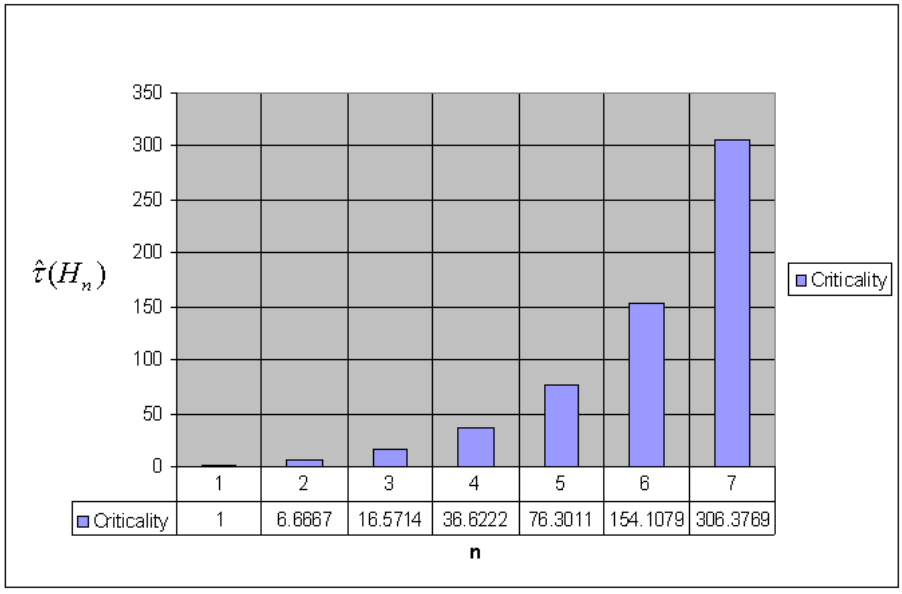


Figure 5.7: Normalized Network Criticality ($\hat{\tau}$) for Hypercube

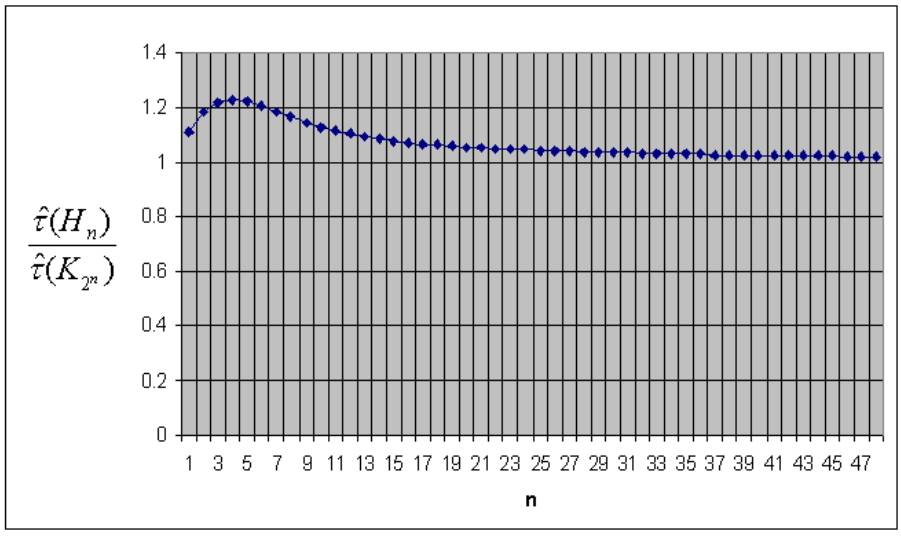


Figure 5.8: The Ratio of Normalized Network Criticality of Hypercube and Complete Graph

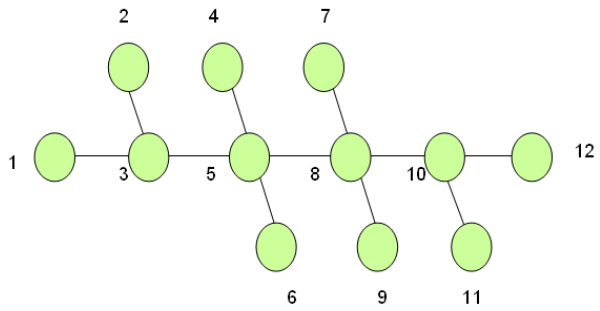


Figure 5.9: Parking-Lot Topology on 12 Nodes

$$\begin{aligned} \frac{\hat{\tau}(H_n)}{\hat{\tau}(K_{2^n})} &= \frac{\frac{n}{(1-\frac{1}{2^n})C}\sum_{i=1}^{n}\frac{C(n,i)}{i}}{\frac{2(2^n-1)}{C}} \\ &\rightarrow \frac{n}{2^{n+1}}\sum_{i=1}^{n}\frac{C(n,i)}{i} \end{aligned} \tag{5.75}$$

Fig. 5.8 shows the graphical behavior of equation 5.75 for different values of n. It can be seen that for higher values of n, fraction $\frac{\hat{\tau}(H_n)}{\hat{\tau}(K_{2^n})}$ approaches 1. Note that even for high values of n the difference between the normalized criticality of H_n and K_{2^n} is considerable, although the ratio is decreasing.

5.7.4 Parking-Lot Network

We consider optimization of parking-lot topology on 12 nodes (Fig. 5.9). In this example we assume that all link costs are 1, that is, $z_{ij} = 1 \ \forall \ (i,j) \in E$. We use the semi-definite programming method to find the optimal weight assignment for parking-lot topology. The optimal weight set for parking-lot topology when the total weight is 1000 (*i.e.* $\sum_{(i,j)\in E} w_{ij} = 1000$), is shown in Table 5.1. It is clear that link $(5,8)$ has the maximum weight (intuitively because its load is more than other links).

The optimal weight assignment for parking-lot can also be found from equation 5.69 (note that parking-lot topology is a tree). We have provided the value of n_{ij} for all the links of parking-lot topology in table 5.2. By substituting these n_{ij}'s in equation

Link	Optimal Link Weight
(1,3)	77.2654
(2,3)	77.2654
(3,5)	121.0498
(4,5)	77.2656
(5,6)	77.2653
(5,8)	**139.7768**
(7,8)	77.2655
(8,9)	77.2656
(8,10)	121.0496
(10,11)	77.2655
(10,12)	77.2655

Table 5.1: Optimal Weights for Parking-Lot

5.69, we reach at the optimal weight assignment of table 5.1.

5.7.5 Trap Network

Our next example is the trap network introduced in chapter 3, example 3.2.4. We use the semi-definite programming approach to find the optimal weight assignment for trap network. At first, we consider the special case of equal link costs. Let $z_{ij} = 1 \; \forall \; (i, j) \in E$. The optimal weight matrix W_{opt} for the trap network (Fig. 3.4) is

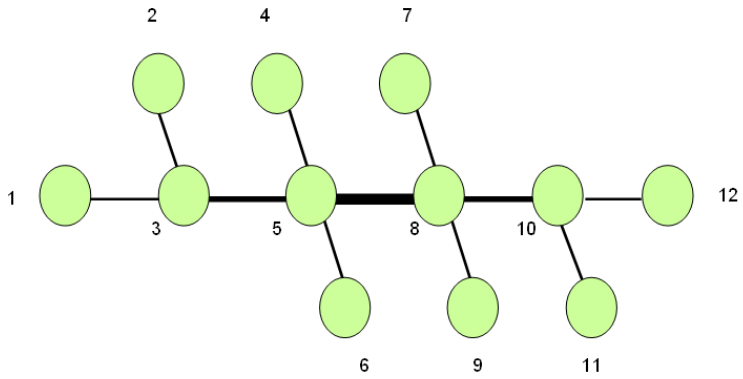


Figure 5.10: Optimal Parking-Lot Topology

Link	n_{ij}	Link	n_{ij}
(1,3)	11	(7,8)	11
(2,3)	11	(8,9)	11
(3,5)	27	(8,10)	27
(4,5)	11	(10,11)	11
(5,6)	113	(10,12)	11
(5,8)	**36**		

Table 5.2: n_{ij} for Parking-lot Topology

given in the following.

$$W_{opt} = [w_{ij}] = \begin{pmatrix} 0 & 135.2401 & 157.9781 & 0 & 0 & 0 \\ 135.2401 & 0 & 0 & 157.9785 & 0 & 0 \\ 157.9781 & 0 & 0 & \mathbf{97.6082} & 157.9764 & 0 \\ 0 & 157.9785 & \mathbf{97.6082} & 0 & 0 & 157.9779 \\ 0 & 0 & 157.9764 & 0 & 0 & 135.2408 \\ 0 & 0 & 0 & 157.9779 & 135.2408 & 0 \end{pmatrix}$$

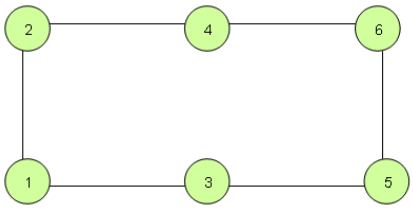


Figure 5.11: Optimized Trap Topology

The minimum weight is assigned to link $(3,4)$. Now we increase the cost of this link to 5, that is $z_{34} = z_{43} = 5$. The new optimal weight matrix would be W'_{opt}:

$$W'_{opt} = [w'_{ij}] = \begin{pmatrix} 0 & 166.6642 & 166.6681 & 0 & 0 & 0 \\ 166.6642 & 0 & 0 & 166.6662 & 0 & 0 \\ 166.6681 & 0 & 0 & \mathbf{0.0000} & 166.6671 & 0 \\ 0 & 166.6662 & \mathbf{0.0000} & 0 & 0 & 166.6691 \\ 0 & 0 & 166.6671 & 0 & 0 & 166.6652 \\ 0 & 0 & 0 & 166.6691 & 166.6652 & 0 \end{pmatrix}$$

One can see that the weight of link $(3,4)$ is now changed to zero, that is, link $(3,4)$ is effectively down. This means that the topology of the trap network is changed. In fact, if we set $w_{34} = w_{43} = 0$ in the optimization problem, and if we use the equal cost for all the links, the optimal weight matrix would still be W'_{opt}. The new topology of the network is shown in Fig. 5.11.

5.7.6 Kleinrock's Network

In the following example our proposed optimal weight assignment method is compared with Kleinrock's method for capacity assignment [104, 87] and Meister's extension [105]. We use the telegraph network of Fig. 5.12 from Kleinrock's book (see [104], pp. 22-23). All of the link cost factors are assumed to be equal to one, that is,

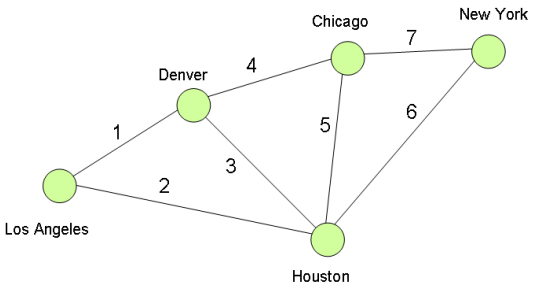


Figure 5.12: Kleinrock's Network

$z_{ij} = 1 \ \forall (i,j) \in E$. Kleinrock's method finds capacities of the links in such a way to minimize the average delay of the network under the independence assumption and when the link loads are known.

One problem with Kleinrock's approach is that it assigns very long delays to the links with small loads. Meister's method is an alternative approach which assigns equal delays to all the links, of course at the expense of a large deviation from optimal average network delay that can be achieved by Kleinrock's solution.

The proposed solution in this paper assigns capacity of the links in a way to balance the individual link delays so as to have acceptable link delays while still we have a good average network delay. Table 5.3 shows the capacity assigned to the links using all the methods. The second column of table 5.3 shows the individual link loads. Columns 3, 4, and 5 show the optimal capacity assignment using Kleinrock's method, Meister's method, and our proposed method (which we call it criticality method) respectively. The minimum average network delay for these methods are given in second column of table 5.4. The third column also shows the value of network criticality. In the criticality method we actually optimize the robustness (not the average delay as it is the case in Kleinrock's and Meister's method), therefore it is not surprising to see that the average delay obtained by criticality method is between two extremes of Kleinrock (to minimize the average network delay) and Meister (to minimize the maximum link

Link	Load	Kleinrock	Meister	Criticality Method
1	3.15	27.93	27.00	29.63
2	3.55	29.85	27.40	33.31
3	0.13	5.16	23.98	12.67
4	3.64	30.28	27.49	32.95
5	0.82	13.46	24.67	13.36
6	3.88	31.38	27.73	33.64
7	9.95	53.99	33.80	36.43

Table 5.3: Capacity Assignment using 3 Different Methods

delay).

Table 5.5 shows individual link values. Kleinrock's method assigns very large delay to link 3 because the demand on link 3 is much less than other links. Meister's method assigns equal delays to all the links. This resolves the issue with Kleinrock's method, but introduces a fairness problem. In our proposed method, the link delays are not equal to allow for fairness based on the demand for each link, and at the same time the individual link delay are kept in a reasonable range.

These examples are simple cases of network design where the goal is to find optimal link weights for a network in order to minimize the network criticality. Note that the case of $w_{ij} = 0$ is also included in the weight assignment problem which leads to the change of topology. Semi-definite programming method can also be used to solve the optimization problem when more constraints are added to problem 5.59. It is also possible to design a numerical iterative algorithm to solve optimization problem 5.59 using standard methods of numerical calculus. For example, theorem 5.5.1, lemma

Method	Average Network Delay	Network Criticality
Kleinrock	44.72	1.06
Meister	55.01	0.80
Criticality Method	49.30	0.56

Table 5.4: Average Network Delay and Network Criticality using Different Methods

5.2.2, and lemma 5.2.4 provide us with enough information to apply Newtonś method to problem 5.59 and find an iterative algorithm to calculate the optimal weights as well as to design distributed traffic engineering algorithms for AutoNet.

Network planning (replanning) in the long-term loop of AutoNet is based on solving an optimization problem similar to 5.59. The constraints of problem 5.59 may be added depending on the type of requests and contracted SLAs with costumers, but as long as positive semi-definiteness is preserved, the method will be exactly the same. In summary, the short-term loop of AutoNet in Fig. 3.1 (and Fig. 3.2) is controlled by modified RW-PCR algorithm, and the long-term loop solves the optimization problem 5.59.

So far, we have discussed algorithmic aspects of AutoNet and provided the details of our traffic management methods as well as network dimensioning (re-dimensioning) strategies. In the following chapter we focus on the architectural aspects of AutoNet and describe the requirements to realize different blocks of AutoNet illustrated in Fig. 3.2.

Link	Kleinrock	Meister	Criticality Method
1	40.36	41.93	37.76
2	38.02	41.93	33.60
3	198.67	41.93	79.71
4	37.54	41.93	34.12
5	79.10	41.93	79.71
6	36.36	41.93	33.60
7	22.71	41.93	37.76

Table 5.5: Individual Link Delays using 3 Different Methods

Chapter 6

AutoNet: Autonomic Network Control and Management System

This chapter is dedicated to the architecture of AutoNet. In chapter 3 we introduced the conceptual architecture of AutoNet. Here we try to shed more light on the building blocks of AutoNet and their interactions. The concept of virtual networks (VN) and autonomic computing are used to develop different blocks of AutoNet.

6.1 Requirements for SLA-based Network Service

In this section, we consider requirements on the SLA-based network service from the perspective of the customers and the service provider. Based on this requirement analysis we design our autonomic network resource management system. A customer is a user of the network resources. The customer can be an enterprise, a content service provider, a 3rd party network re-seller like a VPN provider, and application service provider, e.g. VoIP or IPTV. The network service provider (NSP) owns and operates the physical core network resources. The service instance delivered from the NSP to a customer is a virtual network (VN). The virtual network (VN) is an abstraction of a

physical network that consists of a subset of network resources.

6.1.1 Customer Requirements

Customers have the following important requirements:

1. The customer requires the creation of a VN that can deliver a certain level of quality. The metric of VN quality can be end-to-end bandwidth and delay for multiple routes. When a customer negotiates the SLA for a VN, the VN topology may or may not be specified. If a customer aims to provide service for some other customers, then the customer needs to know the topology of its VN.

2. The customer wishes to monitor the current and historical status of its VN at various times. This VN usage data allows the customer to anticipate future VN usage and to plan for improved utilization.

3. The customer may wish to reconfigure the contracted VN capacity or to remove the VN. For example, suppose a customer provides service for some other customers. The customer may receive requests for service from new clients while it does not have enough resources to address the requirements of new users. Then the contracted SLA should be negotiated with the service provider in order to add more resources.

4. Some customers may choose to create one or more new VNs within their own VN capacity boundary and to resell these to other customers. This reselling activity should be performed independently from the NSP.

5. A VN control and management system should be provided to the customer when a new VN is created. Starting from the physical network infrastructure, a multi-level recursive creation of VNs and associated control and management system should be supported.

6.1.2 Service Provider Requirements

These are important needs of network service providers:

1. Every service provider attempts to optimize the use of its resources (whether physical or virtual) to maximize its service revenue. In this general sense, the NSP that owns the physical network resources has the "root-VN" from which other VNs can be spawned. The VN customer in turn can be a service provider to other customers.

2. The operations expense of a VN should be minimized by automatically and efficiently carrying out the customer or service provider requests. The automatic configuration of VN is very important to reduce network operator errors which are the cause for a significant amount of misuses and faults.

3. The system should provide an autonomic way to handle fault and overload conditions on each VN and on its virtual network resources (VNR). For this autonomic operation, the system should have pre-defined autonomic routines to determine, isolate, and repair faults according to policy.

4. The service provider may wish to provide VN services with different levels of availability (such as 99.999% of availability) to different customers. For this goal, the system should provide appropriate levels of resource redundancy and fast fault recovery mechanisms.

5. When a new VN is created, the system should select an appropriate topology, link bandwidths and call admission control mechanism.

6. The ultimate goal of the NSP is to maximize its revenue, and to do so, it must decide on the optimal mix of services and prices it should offer to its customers based on the available infrastructure and the forecasted demand.

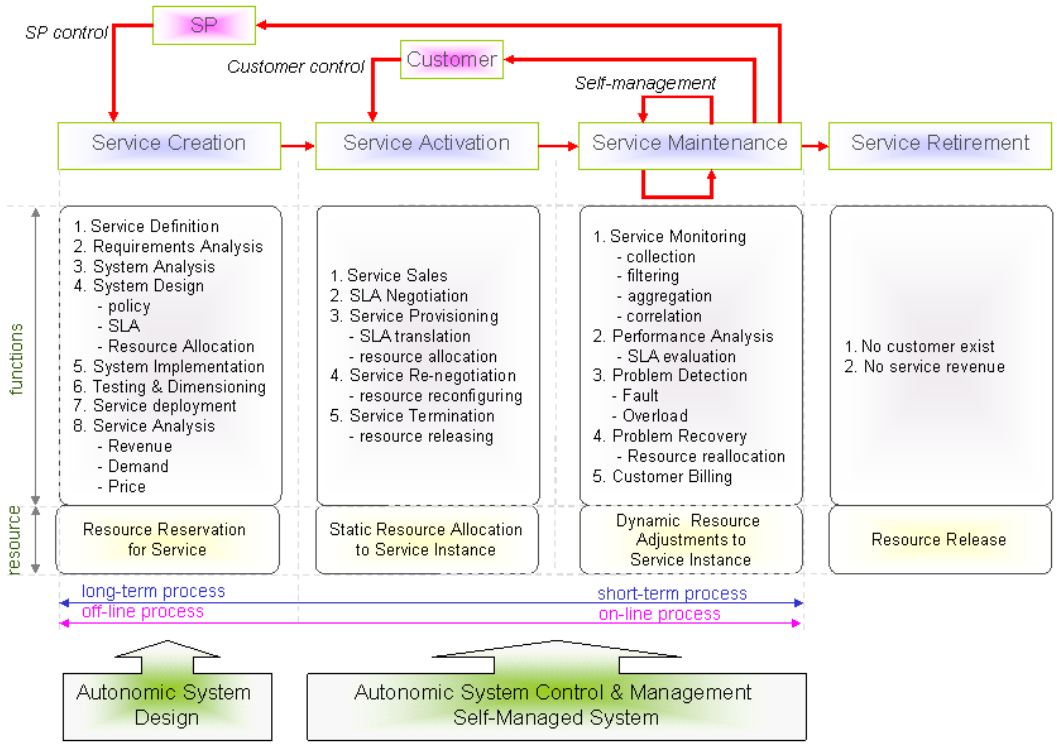


Figure 6.1: Service Life Cycle

6.2 Overall System Architecture

We now describe the overall architecture of AutoNet. The virtual network and autonomic computing concepts are applied to this architecture to make the service provisioning and operation efficient, scalable and cost-effective. In order to find required ingredients of AutoNet, we first define the required life-cycle for a service.

6.2.1 Service Life Cycle and Autonomic Computing

In this thesis the main service is "data transport", however, the following description of service life cycle is general. Fig. 6.1 illustrates a general service life cycle, which has four subsequent stages: service creation, service activation, service maintenance, and service extinction.

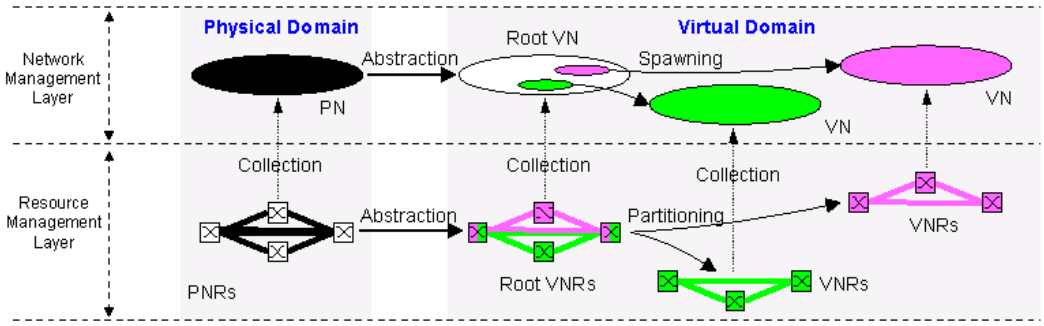


Figure 6.2: A Simple Network Partitioning by VN

The autonomic service provisioning system should carry out the tasks in the stage of the service activation and service maintenance in an autonomic manner. For the autonomic process, the service control and management system should be developed with the autonomic concept in the service creation stage. Therefore, the service provider should consider the autonomic control and management when it develops a new service. With this philosophy, we developed the AutoNet.

6.2.2 Overall Architecture of AutoNet

Virtual network (VN) concept is key to the idea of AutoNet. A VN is the set of network resources that are dedicated in routers and transmission links to one such customer and can be viewed as a subset of the overall network resources ([106, 107]). The partitioning of network resources is made possible by abstracting the set of physical network resources into a set of Virtual Network Resources (VNR). The value of a VNR is specified by a resource quantity (e.g., equivalent bandwidth, protection bandwidth). The allocation of resources to VNs can then be specified in terms of these quantities. A simple example of a VN is shown in Fig. 6.2.

In AutoNet, any managed element is a VN and the management system tries to provide enough resources (according to the contracted SLA) to the clients in a

self-organizing way. We now present a VN-based network resource management architecture which is compatible with the autonomic architecture illustrated in Fig. 3.2. The proposed system attempts to strike a balance between the flexibility of managing VNs separately and the complexity inherent in requiring a network element to interact with multiple managers. The proposed autonomic VN management system is composed of three different autonomic managers: autonomic system re-dimensioning (ASD), autonomic VN manager (AVNM), and autonomic resource manager (ARM), as illustrated in Fig. 6.3. We note that this system follows the traditional manager/agent paradigm along the network and resource management layer. Furthermore, the three autonomic managers are arranged according to the autonomic control loop structure: monitor, analyze, plan, and execute [82, 108].

By monitoring their target managed objects, the autonomic managers diagnose the object status. When a problem is determined, first, the manager tries to localize the problem and repair it by itself. In Fig. 6.3 this local process takes place in autonomic resource managers (ARM). Depending on the nature of the problem this will provide us with self-healing (in case of an error) or self-organizing (re-arrange the resources to attain optimum utilization) or self-configuring (adapt to the other changes). If it cannot handle the problem by itself, the high-level autonomic manager is involved (AVNM in Fig. 6.3).

By this hierarchical autonomic problem handling, the autonomic manager structure can manage and control large networks. The high-level policy from a service provider is deployed automatically through the hierarchical autonomic manager structure and each autonomic manager configures itself based on the given policy (self-configuring), resulting in minimization of human intervention and consequently yields cost effective operation.

Each autonomic manager tries to optimize its corresponding managed objects by forecasting future demand based on policy. The system can also adapt easily to the

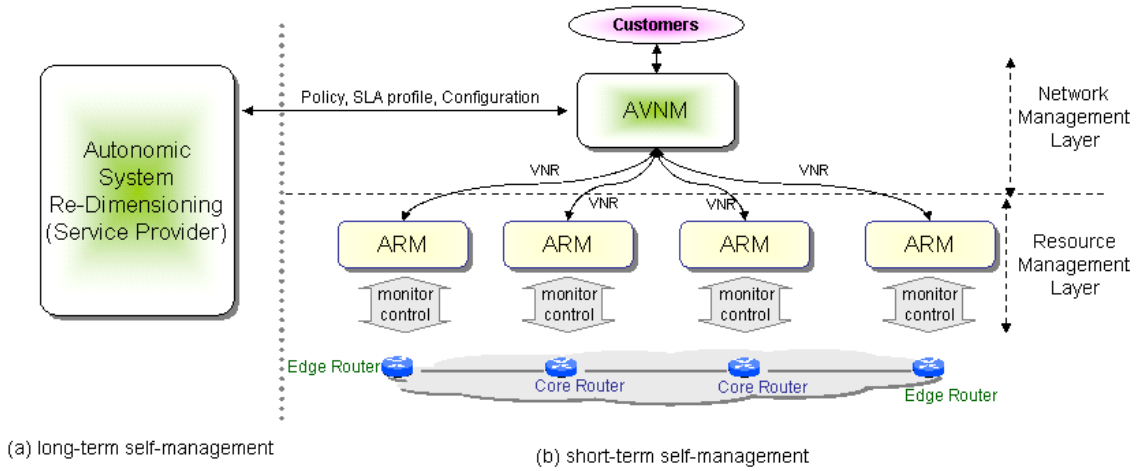


Figure 6.3: Autonomic VN Management System Architecture

changes in network topology by adding new management components in the resource and network management layer.

The ASD is responsible for the long-term loop and is in charge of the business level network analysis and design on a long-term scale as in Fig. 3.2. The role of AVNM is to manage and control the VNs according to the customer SLAs and the service provider's policy. The role of the ARM, on the other hand, is to manage and control the VNRs. A single ARM is responsible for the control of a group of VNRs: a virtual switching resource and multiple virtual link resources connected to it. Customers can create, configure, monitor, and destroy their own VN through the AVNM, and the service provider can configure and monitor its network through the AVNM. Every human action is handled on the AVNM which can separate the customer and service provider's knowledge from the system detailed process. Thus, this architecture can minimize human intervention in the system operation.

Fig. 6.4 illustrates a more detailed view of AutoNet and the interactions with external components (customers, service providers, and physical network resources). The AutoNet provides VN to customers according to the service provider's policy

and customer's quality demand. A customer provided a VN by a service provider, in turn, could be a new service provider that sells some portion of its VN to another customer by VN spawning process. This recursive VN spawning process is one of the key features of our AutoNet system along with the autonomic features, which gives the network re-sellers or service organizers flexibility and controllability in operating their VNs.

The AVNM is responsible for the VN-level control and management, while the ARM is responsible for the element-level resource control and management. In a single VN topology, each switching resource is controlled by one ARM. Multiple AVNMs can exist in our AutoNet in a hierarchical structure like that of the VNs. If there are ten VNs created on a physical core network, there should be ten AVNMs, each of which controls one VN assigned to it. There is no interaction among ARMs in a single VN. The ARMs in different VNs run independently with little interaction, because, when a new VN is spawned from a parent VN, the parent ARM creates a child ARM when the parent AVNM creates the child AVNM.

6.2.3 Spawning in AutoNet

Fig. 6.5 illustrates an example of VN spawning operation and the relationship among AVNMs and ARMs. The informational model in Fig. 6.5 shows that the VN1 spawns the VN2 and the VN2 in turn spawns VN3. The ownership of VN1, VN2, and VN3 is to the SP, CSP (Customer-Service Provider), and CS, respectively. Initially, the SP controls VN1 through the AVNM1, ARM1 and ARM2. In this example, there are two network elements which can be routers or switches in a core network. When CSP wants a VN (VN2 in this example) it contacts AVNM1 and requests a new VN creation with a given SLA. After receiving a VN creation request, the AVNM1 calculates an optimal VN topology and effective bandwidth for each end-to-end route on this topology to create a new VN for the CSP. After creating VN2 for the CSP, the AVNM1

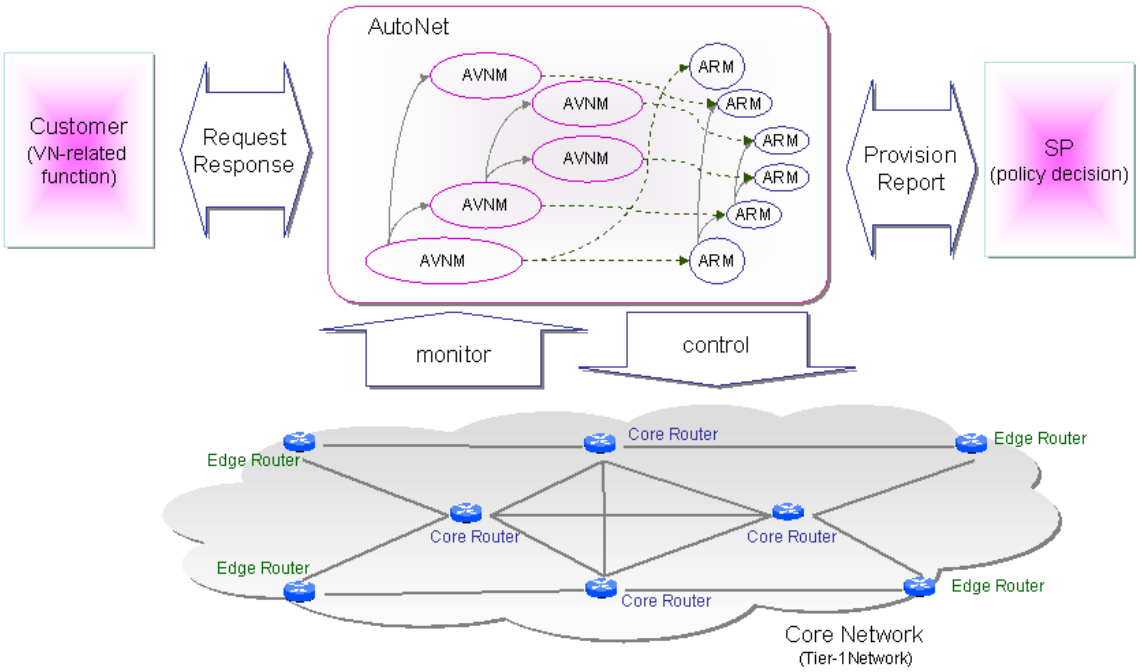


Figure 6.4: Overall Architecture of AutoNet

and its ARMs (ARM1 and ARM2) perform additional actions to provide the CSP with the controllability of VN2. AVNM1 creates AVNM2, and ARM1 and ARM2 spawn ARM3 and ARM4, respectively. With the newly created AVNM2, ARM3, and ARM4, the CSP can control its own VN2. Similarly, CS contacts AVNM2 to create a new VN. AVNM2 and ARM3 spawn AVNM3 and ARM5, respectively, after they create VN3. The recursive spawning of AVNMs and ARMs, allows the customer of a VN to have controllability to its own VN, independently of other VNs and customers. Furthermore, this hierarchical processing of VNs provides scalability and extendibility.

All AVNMs and ARMs are autonomic managers. The autonomic manager handles requests or notifications from external components in an autonomic way. A AVNM communicates with its parent AVNM, its child AVNMs, its owner service provider, customers, and its ARMs. For example, the AVNM2 in Fig. 6.5 receives requests

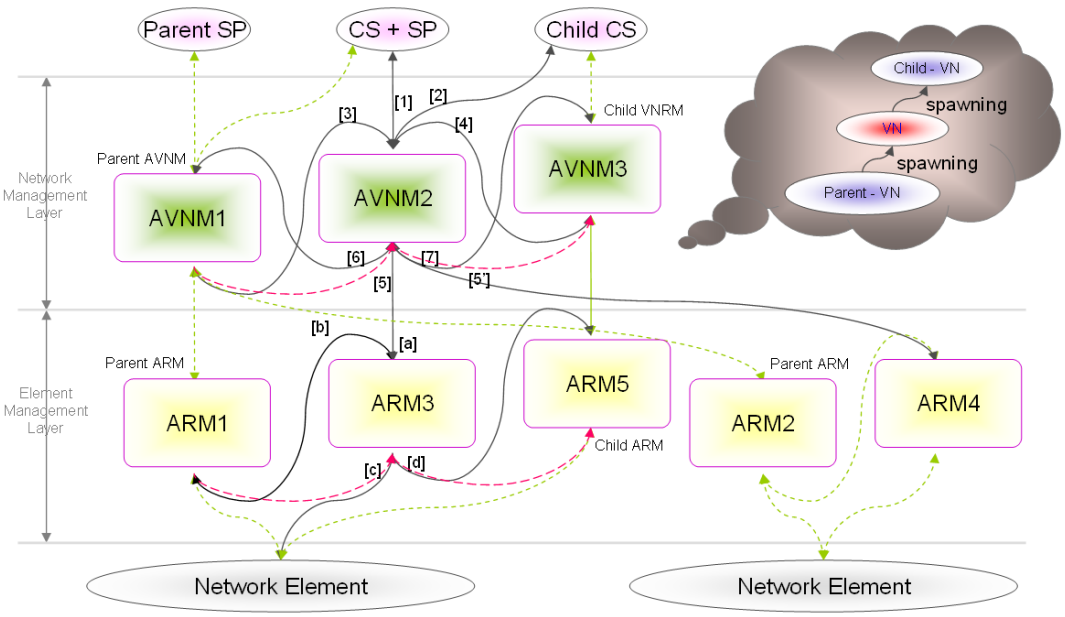


Figure 6.5: Informational Model for AutoNet

or notifications from AVNM1, CSP, CS, ARM3 and ARM4. The contents of external requests are listed in table of Fig. 6.6, which is not exhaustive but important.

In case of ARMs, the external components that generate requests or notifications are the AVNM, the parent ARM and the physical network element. For example, ARM3 in Fig. 6.5 receives the requests and notifications from AVNM2, ARM1, and the physical element 1.

The high-level policy from a service provider is deployed automatically through the hierarchical autonomic manager structure and each autonomic manager configures itself based on the given policy (self-configuring) and processes requests from external components based on the policy. This minimizes human intervention and leads to cost effective operation. A complete set of events between AVNM and ARM is shown in Fig. D.5 of Appendix D.

No	From	To	Operation	Etc
[1]	SP	AVNM	VN policy setup request, VN reconfiguration request, VN removal request	
[2]	CS	AVNM	CS subscription, CS SLA negotiation, Child VN creation request, Child VN reconfiguration, Child VN removal	
[3]	Parent AVNM	AVNM	AVNM creation, VN status notification (in case of fault/overload in parent level detected)	
[4]	Child AVNM	AVNM	Child VN reconfiguration2 request, Child VN removal2 request	
[5]	AVNM	ARM	ARM policy setup, VNR partitioning request. VNR reconfiguration request, VNR removal request, VNR monitoring request	
[6]	AVNM	Parent AVNM	VN reconfiguration2 request, VN removal2 request	[4]
[7]	AVNM	Child AVNM	Child AVNM creation, Child VN status notification	[3]
[a]	AVNM	ARM	ARM policy setup, VNR partitioning request. VNR reconfiguration request, VNR removal request, VNR monitoring request	[5]
[b]	Parent ARM	ARM	ARM creation, VNR status notification2	
[c]	ARM	Physical NE	NE configuration, NE status monitoring	
[d]	ARM	Child ARM	Child ARM creation, Child VNR status notification2 (in case parent level reconfiguration performed)	[b]

Figure 6.6: Requests from Different Blocks of AutoNet

6.3 Detailed Architecture of AutoNet

We now present the detailed architecture of AVNM and ARM, the two types of autonomic managers in the AutoNet. We describe the functional building blocks of the AVNM and ARM and show an example of autonomic operation of AutoNet.

6.3.1 Autonomic Manager Functional Blocks

First, we define generic functional building blocks for an autonomic manager as an extension of the IBM autonomic manager structure [82]. To achieve complete autonomic computing for a service, the four parts of the Fig. 6.7 should be executed automatically in each autonomic manager: customer control, policy control, service activation, and service maintenance. These four parts are tightly coupled and communicate to each other to achieve the ultimate goal: self-management of a service. This concept can be

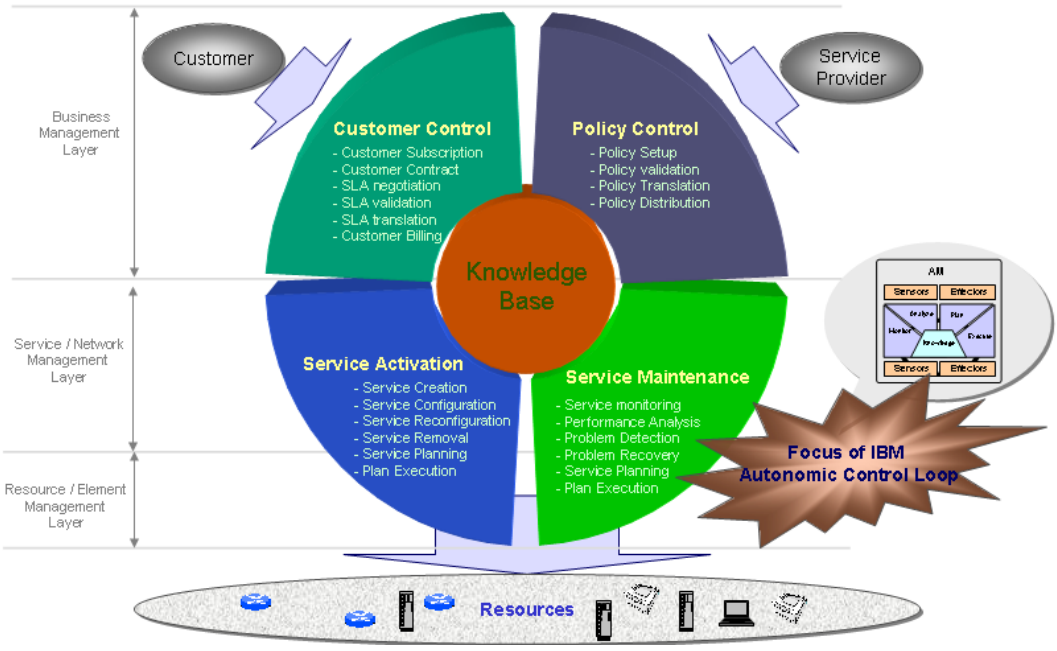


Figure 6.7: The Functional Building Blocks of an Autonomic Manager

applied to all kinds of services, including application-layer services and network-layer services.

The service provider uses the policy control building block to set up policy for the other three functional building blocks. The customer control building block is in charge of the interaction with customers. The customer contacts the system through the customer control block and negotiates the service quality. Based on the SLA information, the service activation building block creates a new service instance for the customer. The service maintenance building block controls and manages the resources during the lifetime of each service instance (problem detection, problem recovery, etc). The maintenance outcomes are sent to the customer control for customer care functionality, such as billing. There is some functional overlap among these parts. For example, the service planning to find optimal resource allocation could be processed in the service activation and service maintenance building blocks.

The IBM autonomic control loop architecture [82] mainly concentrates on the service

maintenance part of the Fig. 6.7, and gives less attention to the interactions with a service provider and customers of the autonomic system. Also, the deployment of services and the creation of new service instances are not considered.

6.3.2 Detailed Architecture of AVNM

Fig. 6.8 illustrates the detailed ten functional building blocks of the AVNM. The AVNM has three different knowledge bases: PIB (Policy Information Base), CIB (Customer Information Base), and VNIB (Virtual Network Information Base), which store static and dynamic information about policy, customer, and VN, respectively. The Policy Manager is responsible for policy translation, policy distribution, and policy validation. It receives the SP's policy and translates it into VN and VNR policy. The VN policy is stored in the PIB and VNR policy is distributed to all ARMs through the Resource Manager. The appropriate policy form for autonomic manager is one of our and others on-going research investigations [109]. In addition to the policy operation, the Policy Manager handles all other requests from the service provider, e.g. VN reconfiguration and VN removal requests. The Request Manager, on the other hand, handles all the requests from customers.

The Request Manager is responsible for customer contact and SLA negotiation. To create a new VN, a customer should subscribe itself to the AVNM and specify VN quality requirements through the Request Manager. The Request Manager also validates the customer VN requirements based on the pre-defined policy and its current VN status, and if appropriate it sends a VN creation request to the VN Manager.

The VN Manager is responsible for the creation, modification, and removal of child VNs. On reception of a VN creation request, the VN Manager makes a topology request to the Topology Manager to find an appropriate VN topology. The VN Manager then makes a resource reservation request to the Resource Manager. In addition, the VN Manager handles the VN re-configuration and removal request from the Request

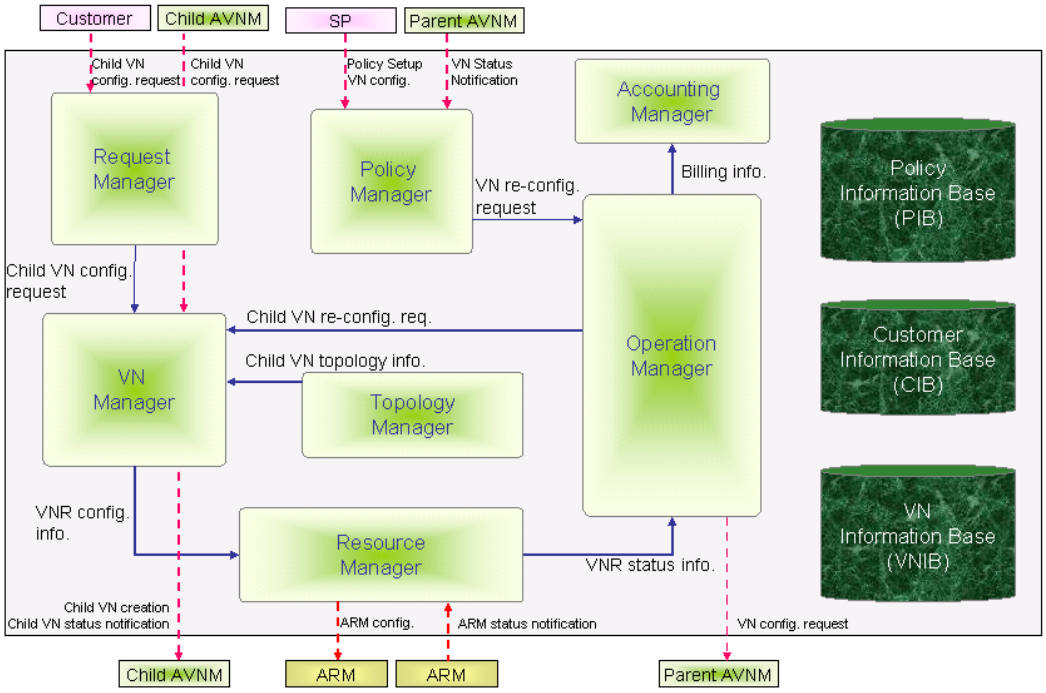


Figure 6.8: The Functional Building Blocks of AVNM

Manager and the child AVNM. The VN Topology Manager is responsible for creating a set of optimal routes and calculating the corresponding effective bandwidth when a new VN is created, or when a VN is reconfigured. Much research [41, 110, 34] has been devoted to route-level QoS, but extensions are required for VN-level QoS guarantees. The goal of VN manager is to find optimal VN allocation from the given network resources that maximize the resource utilization and service revenue.

The Resource Manager is responsible for the communication between the AVNM and the ARMs. The resource manager translates the VN spawning message into a VNR partitioning message and distributes it to the corresponding ARMs to create a new VN. It also sends monitoring requests to the ARMs and receives status information from them. The Operation Manager is responsible for the analysis of each child VN and the overall VN status. It determines VN level problems and sends a VN reconfiguration message to the VN Manager. In addition, it handles the SLA monitoring. The SLA

analysis result is stored in the CIB, which is used for customer billing by the Accounting Manager.

The AVNM implements the autonomic control loop that involves the four main components: Resource Manager, Operation Manager, VN Manager, and Topology Manager. The Resource Manager does monitoring; the Operation Manager does monitoring and analysis; the VN Manager and Topology Managers do planning; and the Resource Manager performs execution. Furthermore, the seven functional components follow the four autonomic functional components described in Fig. 6.7.

6.3.3 Detailed Architecture of ARM

The ARM is composed of six functional building blocks, as illustrated in Fig. 6.9, which interact with each other based on ARM policy. The Request Controller receives VNR partitioning/reconfiguration requests and sends them to the VNR Controller after validating the request. In addition, the Request Controller handles the ARM policy setup request from the AVNM. The Request Controller is in charge of customer control and policy control in the autonomic functional building block in Fig. 6.8.

The VNR Controller plans the operation of each virtual resource for incoming traffic. For example, the methods of call admission control at edge routers and alternative route selection at edge and core routers in case of congestion are decided and optimized in the VNR Controller. Furthermore, the VNR Controller creates a new ARM when a new VNR is created for a new VN and it sends VNR modification alarm reports to a child ARM when its VNR status has changed. The Operation Controller determines VNR level problems and tries to fix them before sending alarms to the AVNM. The VNR status data is delivered from the Resource Controller of the same ARM and the VNR controller of the parent ARM. The Resource Controller is responsible for the control of physical network equipment. The serialization of operations from multiple ARMs is another on-going investigation in the context of IP/MPLS networks.

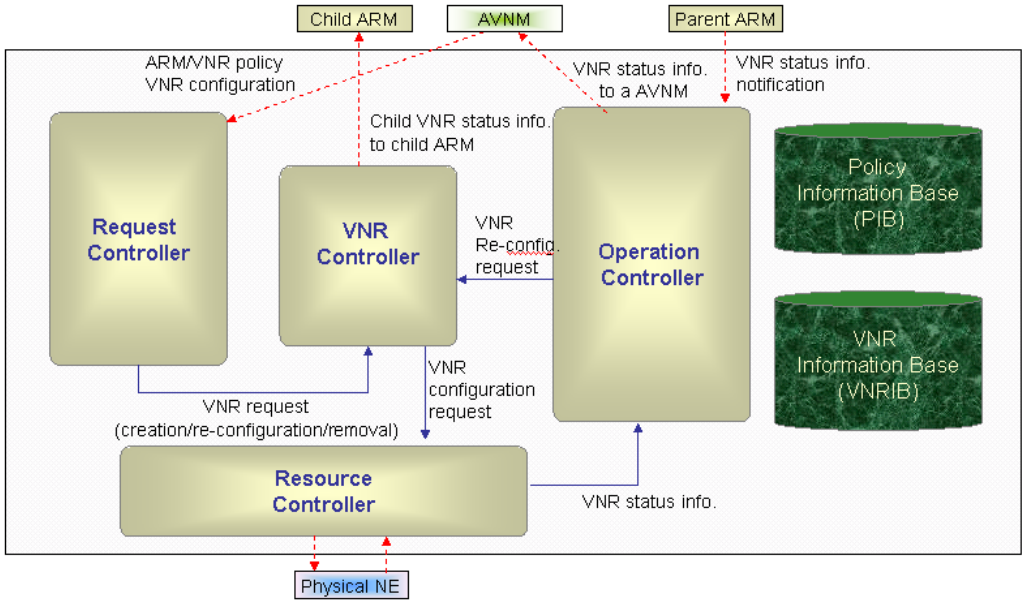


Figure 6.9: The Functional Building Blocks of ARM

The Resource Controller, Operation Controller, and VNR Controller form the main autonomic control loop in the ARM. The ARM periodically retrieves status data of the VNR through the Resource Controller. When any fault is detected the Operation Controller tries to resolve the problem through the VNR controller. The Operation Controller coordinates the sequence of VNR recovery when more than one VNR are in an unstable state. The VNR Controller finds the optimal solution for the given problem based on the VNR policy and configures the physical network device through the Resource Controller. If the problem cannot be resolved by ARM itself, the Operation Controller sends an alarm notification to the corresponding AVNM.

6.4 Autonomic Nature of AutoNet

In the final section of this chapter we show how the autonomicity is provided in AutoNet. To do so, we will consider AutoNet as a system to which external disturbances are applied. Our goal is to show that regardless of the disturbances exerted on

AutoNet, it handles them without human intervention. The players interacting with AutoNet operation are the customers, service providers, resources, and demands. We argue that regardless of the effects of the disturbing elements on AutoNet, the events are handled autonomically.

Service providers interaction is mainly the manual creation of new policies or updating existing policies. The policies entered by the SP manually through the Policy Manager are changed to the appropriate format, and are stored in the PIB. These policies are later used to automate the operation of AutoNet (self-configuring).

Customers interaction involves SLA Agreement by filling the required information. The information entered by the customer in the SLA Agreement is forwarded to the SLA Translator, which creates runtime policies in the appropriate format and stores them in the PIB (Policy Information Base). These policies regulate the creation of customer VN, its re-configuration in case of unsatisfactory performance detected by the Operation Manager, its removal in case customers terminate the service, and its resource allocation in case of problems such as failures or congestion in the underlying resources are detected by the Operation Manager. Operation Manager then takes appropriate actions remedy the problem (self-healing) and re-optimize the operation of the whole system in response to the customer's contracted SLA (self-optimizing).

In case of **demand fluctuation** dynamic resource allocation and re-allocation based on problems detected or SLA violations observed, are performed by the Topology Manager. In addition, performance is evaluated periodically by monitoring the resources via Resource Manager and Operation Manager. If this performance is not deemed optimal, reconfigurations can take place to ensure optimal resource use by the Topology Manager. In some cases, predictions of future demands happen and AutoNet can anticipate outages, congestion, or any kind of problems, using the knowledge accumulated in the VNIB as an indicator. These are the self-configuring, self-optimizing, and self-protecting properties of AutoNet.

Resource problems such as Faults, overloads, congestion of resources, detected by analyzing the raw performance data obtained from the Resource Manager. The faults, overloads, and congestion detected from the measurement infrastructure by the Resource Manager, are fed to the Operation Manager. In Operation Manager the problem is evaluated against the contracted SLA to check for any SLA violation, and to recognize the source of the problem. The results from these are fed into the Topology Manager along with relevant policies from the PIB, so that it decides on the appropriate actions to remedy to the situation at hand. The long-term algorithm in Topology Manager analyzes the problem to take appropriate actions in order to prevent future problems, if necessary. These are the self-healing, self-optimizing, and self-protecting properties of AutoNet.

Chapter 7

Extensions and Further Work

There are several avenues for further research on network criticality. In general the research works fall into two main categories: 1. theory, 2. applications. In this chapter we attempt to describe some of the possible extensions in each one of these categories.

7.1 Theory

Theory of the network criticality can be extended in different ways.

7.1.1 Asymmetric Weight Matrix

In our work we supposed that the weight matrix of the network is symmetric. In some practical situations we need to consider the case of asymmetric weight matrices. Chapter 3 of this thesis is not changed even if we have an asymmetric weight matrix, but in chapter 4 we use the symmetry of weight matrix and express the criticality based on the graph Laplacian which is also a symmetric matrix. When the weight matrix is asymmetric, we have two options, we need to either convert the directed graph to an undirected graph with appropriate weight matrix, or we need to use the mathematics of directed graphs [111, 112, 113]. In [111] the Laplacian of a directed graph is defined

as follows. Let $\overrightarrow{\pi}$ be the left eigenvector of transition probability matrix P, that is, $\overrightarrow{\pi} = \overrightarrow{\pi}P$. The Laplacian of a directed graph is:

$$L = I - \frac{\Phi^{\frac{1}{2}} P \Phi^{-\frac{1}{2}} + \Phi^{-\frac{1}{2}} P^t \Phi^{\frac{1}{2}}}{2} \tag{7.1}$$

where Φ is a diagonal matrix whose main diagonal entries are the elements of $\overrightarrow{\pi}$. While the graph is assumed to be directed, the Laplacian (equation 7.1) is still a symmetric matrix. Equation 7.1 can be used as an starting point to extend the results of this thesis to digraphs.

Another version of the Laplacian matrix for digraphs is introduced in [114] which is asymmetric in nature and its definition is similar to the definition of Laplacian matrix for undirected graphs ($L = D - W$). This provides another method to extend the results of this thesis to the asymmetric case.

7.1.2 Loop-Free Random-Walk Betweenness

Our definition of random-walk betweenness in chapter 3 and 4 allows for loops. If we need to find loop-free betweenness of a node k, we have to count the number of random-walks in reverse direction passing the links incident to node k (Fig. 7.1). For every incident link (k, k_i) the average number of walks from node k to k_i is $b_{sk}(d)p_{kk_i}$, and the average number of walks in reverse direction (node k_i to k) is $b_{sk_i}(d)p_{k_ik}$. Therefore, the net number of walks passing node k will be:

$$\begin{aligned} b'_{sk}(d) &= \sum_i (b_{sk}(d)p_{kk_i}(d) - b_{sk_i}(d)p_{k_ik}(d)) \\ &= \sum_i (b_{sk}(d)\frac{w_{kk_i}}{W_k} - b_{sk_i}(d)\frac{w_{k_ik}}{W_{k_i}}) \\ &= \sum_i (\frac{b_{sk}(d)}{W_k} - \frac{b_{sk_i}(d)}{W_{k_i}})w_{kk_i} \end{aligned} \tag{7.2}$$

Now we use equation 4.15 to simplify equation 7.2.

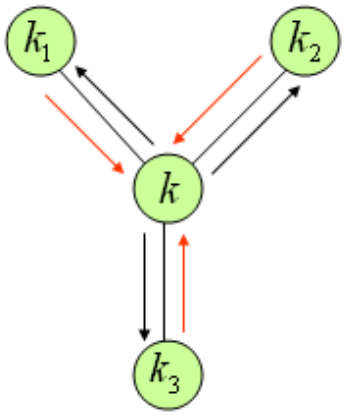


Figure 7.1: Loop-Free Betweenness

$$\begin{aligned} b'_{sk}(d) &= \sum_i (l^+_{sk} - l^+_{sd} - l^+_{dk} + l^+_{dd} - l^+_{sk_i} + l^+_{sd} + l^+_{dk_i} - l^+_{dd}) w_{kk_i} \\ &= \sum_i (l^+_{sk} - l^+_{dk} - l^+_{sk_i} + l^+_{dk_i}) w_{kk_i} \end{aligned} \tag{7.3}$$

Equation 7.3 expresses the loop-free betweenness of node k for source-destination pair $s-d$ based on the Moore-Penrose inverse of the graph Laplacian. This equation can be used as the main definition of the node betweenness and can be synthesized using the same approach we took in chapter 4 and 5.

7.1.3 Extending the Definition of Betweenness

In our work the effect of traffic demand is implicitly applied into the weight matrix of the graph. Equation 3.10 can be used to define another variant of random-walk betweenness that explicitly takes into account the effect of external input rates at each node. Motivated by equation 3.10, we can define this new betweenness measure as follows.

$$b'_k = \sum_d \sum_s \frac{\lambda_s(d)}{\lambda} b_{sk}(d) \tag{7.4}$$

where $\lambda = \sum_d \sum_s \lambda_s(d)$ is the total external traffic rate. b'_k is a weighted average of random-walk betweenness for every source-destination pair $s-d$ over all available

$s-d$ pairs.

A nice research work is to study the behavior of b'_k. The work includes analytical study as well as simulation-based investigation. The goal is to find the attributes of b'_k and propose useful guidelines for network control purposes.

7.1.4 Investigation of the Community of Interest

In the analysis of random-walk betweenness throughout this thesis, we mostly considered the largest possible community of interest where all possible source-destination pairs on a network assumed to be active. The definition of network criticality in equation 4.19 considers all $s-d$ pairs as active source and sinks for traffic. In chapter 5 we saw that this definition of network criticality provides a strictly convex function of weight matrix which leads to the synthesis of network criticality in chapter 5.

In some scenarios we may have a subset of source-destination pairs as active nodes for originating and terminating the network traffic. In these situations the definition of network criticality may include only the active $s-d$ pairs as the CoI (community of interest):

$$\tau = \sum_{sd \in CoI} \tau_{sd} \tag{7.5}$$

Equation 7.5 also defines a convex function of weights, since τ_{sd} is convex, but it is not necessarily a strictly convex function. This means that the optimization problem 5.17 does not have unique solution. The optimization problem for this case is:

$$\begin{aligned} &Minimize && \textstyle\sum_{sd \in CoI} \tau_{sd} \\ &Subject\ to && \textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} = C \quad , C\ is\ fixed \\ & && w_{ij} \geq 0 \ \ \forall\ (i,j) \in E \end{aligned} \tag{7.6}$$

Semi-definite programming technique can still be used to solve problem 7.6. The semi-definite form of problem 7.6 is similar to semi-definite form 5.59, the only differ-

ence is in the objective function of the optimization problem:

$$\begin{gathered} Minimize \quad \sum_{sd \in CoI} u_{sd}^t \Gamma^{-1} u_{sd} \\ Subject\ to \quad \Gamma = L + \frac{J}{n} \\ Tr(Diag(Vec(Z)) \times Diag(Vec(W))) = C \\ \begin{pmatrix} \Gamma & I \\ I & L + \frac{J}{n} \end{pmatrix} \geq 0 \\ Diag(Vec(W)) \geq 0 \end{gathered} \tag{7.7}$$

Optimization problem 7.7 can be solved using standard techniques of semi-definite programming. Since the solution is not unique, we need to choose an appropriate solution from the whole solution-space. Therefore, we need to have an in-depth study on problem 7.6. It seems that we need to add some more constraints to the problem to decrease the solution-space.

7.1.5 SLA-Weight Mapping

The problem of mapping QoS constraints to link weights is not new. In multi constraint problems (MCP), there is usually a vector of weights assigned to each link of the network to quantify QoS constraints. In MCP we try to find a network control strategy to meet these constraints to the extent possible. Jaffe [115] proposed a shortest path algorithm using a linear combination of the link weights. He considered an equivalent link weight that was equal to the sum of the QoS weights for the link. In [88] a more effective method is proposed based on Holders q-vector norm:

$$w_{eff}(l) = (\sum_{i=1}^{m} [\frac{w_i(l)}{L_i}]^q)^{\frac{1}{q}}$$

where $w_i(l)$ denotes i^{th} QoS weight of link l, L_i denotes the maximum value of $w_i(l)$, and $w_{eff}(l)$ denotes the effective weight of link l. When $q \to \infty$, the Holders q-vector

norm is converted to the max operator:

$$w_\infty(l) = \max_{1 \le i \le m}[\frac{w_i(l)}{L_i}]$$

In this thesis we suggested an alternative way to define link weights to include QoS metrics (equation 4.3), but throughout the thesis we assumed that the SLA constraints are already mapped to the weights. One research topic is to investigate the effectiveness of different methods to map SLA parameters to the weight and find an appropriate approach.

7.1.6 Network Criticality and Graph Spectrum

In chapter 4 we saw that network criticality is bounded by the reciprocal of algebraic connectivity (see theorem 4.5.4). As a matter of fact there is a close similarity between network criticality and algebraic connectivity. Table 7.1 summarizes some important properties of τ and λ_2.

Network Criticality (τ)	Algebraic Connectivity (λ_2)
$\frac{\partial \tau}{\partial w_{ij}} = -2n\|\|L^+ u_{ij}\|\|$	$\frac{\partial \lambda_2}{\partial w_{ij}} = \|\|v_2^t u_{ij}\|\|$
$\frac{\partial \tau^2}{\partial^2 w_{ij}} = -2u_{ij}^t L^+ u_{ij} \frac{\partial \tau}{\partial w_{ij}}$	$\frac{\partial \lambda_2^2}{\partial^2 w_{ij}} = 2u_{ij}^t Q^+ u_{ij} \frac{\partial \lambda_2}{\partial w_{ij}}$
Convex	Concave

Table 7.1: Network Criticality vs. Algebraic Connectivity

In table 7.1 v_2 is the Fiedler eigenvector (eigenvector corresponding to λ_2), and $Q = \lambda_2 I - L$. Table 7.1 shows that network criticality and algebraic connectivity have similar functional forms. From robustness point of view the algebraic criticality is investigated in [116, 117]. In [116] the algebraic connectivity is studied in relation to the graphs robustness to node and link failures. Three types of network topologies are considered:

the random graph of Erdos-Renyi, the small-world graph of Watts-Strogatz and the scale-free graph of Barabasi-Albert, and through extensive simulations with the three complex network models, it is shown that the algebraic connectivity is not trivially connected to the graph robustness to node and link failures. The authors have shown that in some cases the speed of increasing the algebraic connectivity is much lower than the speed of increasing node connectivity and it is very dependent on the considered complex network model.

It is worth investigating the behavior of τ when node connectivity increases and compare the results to that of [116, 117]. This investigation can include the analytical study as well as simulation based experiments. An experimental plan for this investigation is provided in section 7.2.7. Furthermore, lemma 4.4.2 shows that the reciprocal of network criticality is proportional to the harmonic average of non-zero eigenvalues of the graph Laplacian . Therefore there may exist some bounds for τ in terms of other eigenvalues of the graph Laplaian. One of the future works is to study these bounds and find the relationship between τ and other eigenvalues of the graph Laplacian.

7.2 Applications

In this section we summarize some of the possible applications of network criticality.

7.2.1 Job Assignment Problem

Consider a network of processors in which the processes enter into the network and they should be assigned to the appropriate processor. The goal is to balance the load among all the processors [118, 119]. This is an example of a job assignment problem where the jobs are the processes that need to be assigned to appropriate agents (processors).

In this thesis we considered the flow assignment problem, where the input demands

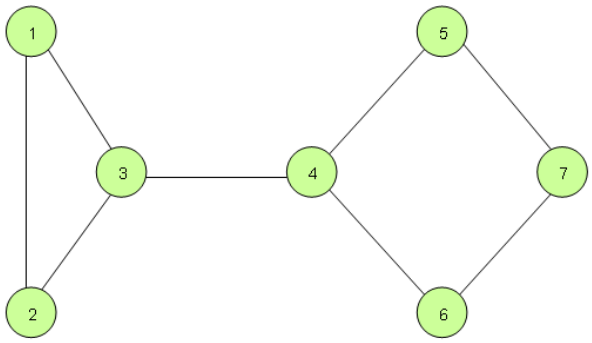


Figure 7.2: Primal Network

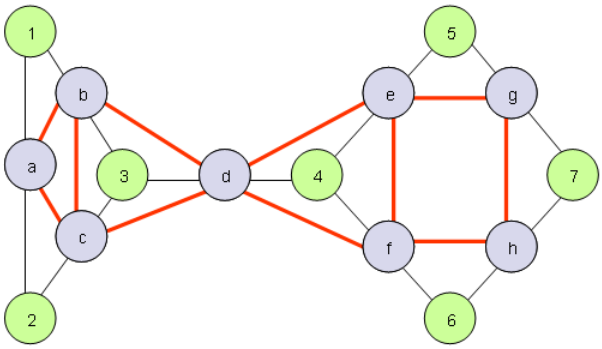


Figure 7.3: Primal vs Linear Dual Graph

are assigned to different links of the network. The flow assignment problem and job assignment problem can be converted to each other using the concept of *Linear Dual* of a graph [120]. Consider the network of Fig. 7.2 as the primal network. Assign a node to each link of the primal network. Connect two nodes if their corresponding links in the primal graph have a common node. The new network topology is called linear dual of the graph and it is shown in Fig. 7.3. Note that linear dual graph is not the same as standard dual of a graph.

Linear dual graph is used in literature for different purposes. For instance, in [121] linear dual graph is used to map a link queue in primal graph to a node potential in linear dual graph. Here we are interested in job assignment problem. The flow

assignment problem in the primal graph is converted to a job assignment problem in the linear dual graph. This means that our results for flow-assignment can be used in a job assignment problem in linear dual graph (and vice versa). It is interesting to see the relationship between the network criticality of primal and linear dual graph. There is a known result about the spectrum of linear dual graph that relates it to the spectrum of the primal one [122]. This can be a good starting point to find the network criticality of linear dual graph.

7.2.2 Improving Service Differentiation

The use of Multi Topology Routing (MTR) for traffic engineering purposes was proposed in [123] based on dividing traffic matrix into smaller slices, each routed on a separate topology the greater the number of slices, the better the performance as it increases the ability to approximate optimal routing. Recently, the use of MTR has been proposed to improve the resiliency of IP routing [124, 125, 126], with different topologies offering backup routes for different failure scenarios. [127] seeks to investigate how routing influences a networks ability to efficiently support different service classes. Of particular interest is the extent to which the ability to route service classes separately is beneficial. This question is explored for a base configuration involving two classes with either similar or entirely different service objectives (cost functions). The contributions of the paper are in demonstrating and quantifying the benefits that the added flexibility of different (dual) routing affords, and in developing an efficient heuristic for computing jointly optimal routing solutions.

The idea of network criticality may be useful to investigate the MTR problem. We can consider a virtual network (VN) assigned to each service level and treat it as an independent network. Each VN has its own resources and network weights, and a network criticality is assigned to each VN. The goal is to solve a joint optimization problem whose final solution is a set of weights for each VN that keeps τ for all VNs

within a pre-defined threshold values. The problem can be formulated with the same approach that we used in chapter 5 to investigate the optimization problem 5.17, but here we have a weight vector assigned to each link of the network and we need to optimize a vector of network criticalities.

7.2.3 Network Simplification

Network criticality can be used to simplify the topology of large networks. Some of the nodes and links in a large graph may not contribute that much in providing the robustness of the network. These are the links and nodes whose removal does not cause any significant change in τ. This provides an approach to reduce the complexity of large network while the robustness properties are preserved. As an example, the nodes with degree one do not significantly affect the criticality of a network. There may exist some more complex structures that do not heavily change the network criticality. One research topic is to find these structures and provide some guidelines for network simplification.

7.2.4 Network Criticality and Ant Colony Algorithm

Ant colony algorithms are a series of routing procedures inspired by the operation of ants in nature [128, 129, 130]. Individual ants seem to move at random, do nothing but wander off, and yet groups of ants can accomplish complex tasks. Somehow, a collective intelligence is formed out of many simple elements which is called swarm intelligence. Each agent (ant) processes a very simple algorithm. The collective outcome realizes a much more complex algorithm. The whole system is distributed and adaptive. Ants cannot see or hear. They only sense the environment, and also the food. Ants cannot talk either, they communicate indirectly through the environment. An ant can leave a trial of pheromones which are materials with particular fragrance.

Ants can smell and sense the pheromones left by other ants, moreover, ants can detect the density of the pheromones.

Ants in networks are emulated by mobile agents. Mobile agents are carried by packets. Special packets can be used as mobile agents (ants). Pheromones pass the information about the length of the path (time) to other ants. The agents can pass the same information to data packets at the nodes. Ants decide based on the density of the pheromones and some probability values. The probability values can be calculated based on the path information and listed in routing tables in the nodes. Starting with a static routing table for each node, every individual routing table stores the probabilities of using the next hops to reach all possible destinations. The sum of all the probabilities at each row should be equal to one.

It appears that the definition of random-walk betweenness can be used to model pheromones. One can consider a random-walk as an ant. Assume that the random-walk (ant) starts at node s and stops at d where the food is located, then the thickness of pheromones on a link can be quantified by random-walk betweenness of the link for source-destination pair $s - d$. In this case the evaporation of the pheromones happens when a demand exits the network. This motivates a new line of research. While the random-walk interpretation of an ant is not new, the notion of τ and its interpretation in ant colony can result in a more understanding of ant's behavior. Since ant colony algorithms have successfully applied to many practical applications (for instance vehicle routing problem), this research can open doors to an in-depth analysis of these problems.

7.2.5 Network Design Problem Revisited

In chapter 5 we introduced the network design problem in the context of network criticality. The solution of optimization problem 5.17 is a set of optimal link weights. These weights in general are a function of link QoS parameters such as delay, packet

loss, available bandwidth and so on. If we just consider the bandwidth as our QoS metric, and assume that the weight of link (i, j) is equal to the available bandwidth of the link, then for unloaded case the optimization problem 5.17 is converted to the capacity assignment problem. For a more general case where links have different QoS parameters, we may need to add more constraints to the optimization problem. The solution of optimization problem 5.17 then provides the optimal weights. Now one needs to map these weights to the link QoS parameters. This reverse map is not unique and needs an in depth investigation to find an appropriate method to do this mapping from link weights to link QoS parameters.

7.2.6 Wireless and Mobility

One can think of different applications of the network criticality in mobile and wireless networks. For instance in mobile wireless networks the topology of the network is dynamically changing. One control problem is to keep the network well-connected. This problem motivates the optimization of algebraic connectivity among different network topologies. Another control problem is to keep the mobile network in least critical situation. Then one needs to find more robust topologies, this motivates the optimization problem to find topologies with least network criticality. Network weights should be defined appropriately. One good choice for the weight of link (i, j) is the reciprocal of the euclidian distance of nodes i and j.

Another line of research in the area of wireless networks can be initiated by defining appropriate link weights to include the effect of interference. Then one can formulate the problem of minimum interference routing as an optimization problem to minimize the network criticality.

7.2.7 Network Criticality of Structured and Random Graphs

In order to compare the robustness properties of different graphs, we need to find the network criticality of different graphs. In the following we provide a list of some important networks accompanied with an experimental plan to assess the robustness of different networks and compare the behavior of the network criticality and algebraic connectivity. This study considers the following networks types.

1. Type 1: Networks with given degree distribution (power law is an example of these networks)
2. Type 2: Networks with given link weight distributions (we assume that the network topology has a fixed number of nodes and randomness is only in weight distribution).
3. Type 3: Networks with given betweenness distribution
4. Type 4: Structured construction of recursive networks using Cartesian and Kronecker products.
5. Type 5: Erdos-Renyi random graphs
6. Type 6: Scale-free networks
7. Type 7: Small-world networks
8. Type 8: Networks with given node/link connectivity (and minimum degree) distribution.
9. Type 9: Hierarchy of networks (type 1-7)

Here is an experimental plan to assess the robustness of these networks.

1. Find and plot the distribution of criticality and algebraic connectivity for networks of type 1-7.

2. Find the distribution of criticality and algebraic connectivity for hierarchical networks. Is there any relationship between criticality and algebraic connectivity of the final network and its ingredients?

3. Investigate the behavior of criticality and algebraic connectivity when the node/link connectivity changes (type 8).

4. Investigate the scalability issues in structured growing networks.

5. Investigate the distance variations in different network types and its effect on criticality and algebraic connectivity.

6. Compare different networks types (their behavior with respect to criticality and algebraic connectivity). This results in a better understanding of the properties of algebraic connectivity and network criticality.

7. Investigate the behavior of network criticality versus other Laplacian eigenvalues of the graph.

7.3 Conclusions

In this thesis, we addressed issues related to the management of backbone (core) transport networks, namely the algorithmic architecture of such network management systems. A main concern in core networks is the existence of unanticipated events in the system, such as traffic surges, a sudden topology change, or changes in active sources (sinks) for traffic. A core network should be able to smoothly react to such unwanted events so as to provide reliable data delivery service for the customers according to the service level agreements (SLA).

The majority of traffic control systems in use by service providers so far are manually configured by human intervention. The continual growth in traffic volume, diversity, and heterogenous requirements make it impossible to continue working with

the present network management systems. Automated service and network management are essential to creating and maintaining a flexible and agile service/application delivery infrastructure that also has much lower operations expense than existing systems.

In this thesis we proposed the AutoNet, a management framework for data delivery over core networks. We argued that the traffic engineering system requirements can be met by a self-management system based on autonomic computing. We gave a conceptual design of our autonomic traffic engineering system, AutoNet, but we focused on a set of essential graph theoretic algorithms that provide the means for adaptive management required by the autonomic system.

The theory of evolution motivated the conceptual idea underlying AutoNet. Evolutionary processes are among self-organizing systems by nature. Darwin's theory describes the process of natural selection by which slight variations, if useful, are preserved. Every process has a survival value as a result of natural selection that quantifies its overall sensitivity or robustness to the external variations. In this thesis we looked for an appropriate survival value for communication networks that indicates how adaptable a system is to unexpected events.

In any network, from small designed networks, to large-scale social networks, and even to the Internet, connectivity is a crucial factor as it is essential for communication. Therefore, the first parameter to consider as a candidate for "survival value" is the connectivity of the graph. In the first part of this thesis we proposed some metrics, by extending some ideas from graph-theory, to model the robustness of a network. This is the first step toward defining an appropriate survival value for communication networks. We defined Link Criticality Index (*LCI*) as the deterministic betweenness of a link per unit of available link bandwidth (betweenness of the link over its available bandwidth). We proposed Path Criticality Routing (PCR) algorithm based on evaluating the *LCI* of different links of the network. This algorithm tries to find the least

critical path. The success of PCR algorithm encouraged us to study the behavior of *LCI* analytically. This study is the subject of the second part of this thesis.

We extended the idea of *LCI* and defined node/link criticality as the probabilistic betweenness of the node/link per unit of node/link weight (random-walk betweenness of the node/link over its weight). We showed that the criticality of a node/link is independent of the position of the node/link in the network, and it is only a function of link weights. We termed the fraction of node/link betweenness over the node/link weight "network criticality" and investigated its properties. The network criticality is a global metric quantifying the robustness of the network, therefore, we used network criticality as the survival value for the networks. Any communication network should evolve in a way that minimizes the network criticality.

While Darwin's theory does not consider any "final target" for the evolutionary changes in the nature, viewing survival as the network management goal can lead to an implicit optimization problem. The optimization must address the real-time efficiency and performance of the whole network as a short-term goal, while striving to maintain and improve the survival value of the network as a long-term goal. This is the subject of the third part of the thesis. We investigated the problem of optimizing the survival value (network criticality) under some constraints. We proved that network criticality is a strictly convex function of link weights, and investigated the problem of minimizing network criticality as a convex optimization problem. This led to some guidelines for designing more precise traffic management methods (improving PCR algorithm), as well as directions for network planning problem.

In the final part of the thesis, we returned to the architecture of AutoNet and provided details of its necessary building blocks. Our emphasis in this section of the thesis was to show how the concepts of autonomic computing and virtual networks can be used to build autonomic networks capable of self-optimizing, self-configuring and self-healing.

This research work is just in its early stage of development. There are definitely much more research questions that need to be addressed. While most of the work over the last years has focused on frameworks and models, this thesis attempted to tackle the algorithmic aspect of the autonomic promise in the telecommunications world. An abundance of work remains to be done on these issues, and will definitely constitute a major research area for years to come.

Appendix A

Graph Laplacian

In the following we provide a short review of the graph Laplacian matrix [89, 90, 91, 92].

A.1 The Definition of Graph Laplacian for Simple Graphs

Given a simple graph $G = (N, E)$ with n vertices, we define the adjacency matrix $A(G)$ to be the n-by-n matrix with entries a_{ij} such that a_{ij} is 1 if (i, j) is an edge, and a_{ij} is zero otherwise. That is:

$$a_{ij} = \begin{cases} 1 & if \ \ (i,j) \in E \\ 0 & otherwise \end{cases}$$

Example A.1.1 *For the graph of Fig. A.1 the adjacency matrix is the following.*

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

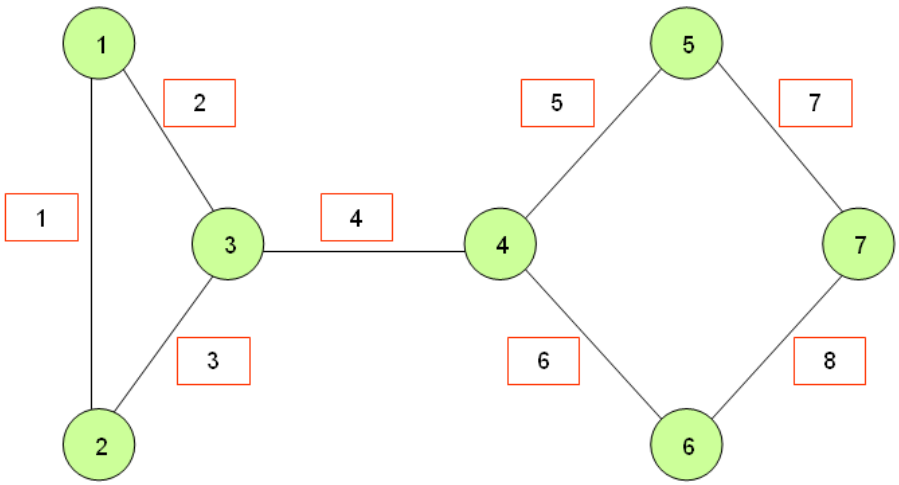


Figure A.1: Fish Network Topology (not weighted)

We define the degree matrix $D(G)$ to be the n-by-n diagonal matrix such that the main diagonal entry d_{ii} is the degree of node i. That is

$$d_{ij} = \begin{cases} d_i & if \;\; i = j \\ 0 & otherwise \end{cases}$$

where d_i denotes the degree of node i.

Example A.1.2 *The degree matrix of the fish network (Fig. A.1) is given in the following.*

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Now we define the Laplacian $L(G)$ to be the n-by-n matrix with diagonal entries d_i and off-diagonal entries l_{ij} such that d_i is the degree of vertex v_i, and l_{ij} is -1 if (i, j) is

an edge, and l_{ij} is 0 otherwise. That is

$$l_{ij} = \begin{cases} d_i & if \;\; i = j \\ -1 & if \;\; (i,j) \in E \\ 0 & otherwise \end{cases}$$

Observe that $L(G)$ is a symmetric matrix. It is easy to see that the Laplacian matrix can be obtained from the following equation: $L = D - A$.

Example A.1.3 *The Laplacian matrix of the fish network is as follows.*

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

Now let us define another matrix which encodes information from a graph. For a simple graph $G = (N, E)$ with n vertices and m edges, define the *incidence matrix* $U(G)$ to be the n-by-m matrix defined as follows. Each column of U corresponds to an edge (i, j) of G. In that column, put a "1" in the ith row, a "-1" in the jth row, and zeros everywhere else. (Notice that we equally well could have put a -1 in the ith row and a "1" in the jth row. You could devise some rule for which row gets the 1 and which gets the "-1", but for our purposes it doesn't matter.) For the graph above, the incidence matrix is

$$U = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

Proposition A.1.4 *$L(G) = U(G)U(G)^t$ (where ^{t} denotes transpose).*

Proof An entry of $U(G)(U(G))^t$ is the inner product of two rows of $U(G)$. The inner product of a row with itself is simply the number of nonzero entries in that row, which is equal to the number of edges incident to that vertex. The inner product of two different rows of $U(G)$ is 0 if there is no edge between those two vertices, and is -1 if there is an edge. So $U(G)U(G)^t$ is in fact the Laplacian. ∎

Observation A.1.5 *If we let u_i be the i^{th} unit vector, then $u_{ij} = u_i - u_j$ is in fact the l^{th} column of U, where $l = (i, j)$.*

Take a vector $x = (x_1, x_2, \ldots, x_n)^t \in R^n$. Then we have

$$x^t L x = x^t B \cdot B^t x = (x^t B) \cdot (x^t B)^t = \sum_{(i,j) \in E(G)} (x_i - x_j)^2$$

Let us list the facts we know about L(G).

$L = L^t$

$L = BB^t$

$x^t L x = \sum_{(i,j) \in E(G)} (x_i - x_j)^2, \forall x \in R^n$

All eigenvalues of L are real and non-negative. In fact, L is a positive semi-definite matrix since $x^t L x \geq 0 \; \forall \; x \in R^n$

A.2 Graph Laplacian for Weighted Graphs

The definition of graph Laplacian can be extended for weighted graphs as follows. Consider a graph $G(N, E, W)$, where N and E denote the set of nodes and links respectively, and W denotes the n-by-n matrix of link weights. Define the Laplacian $L(G)$ to be the n-by-n matrix with diagonal entries $W_i = \sum_k w_{ik}$ (we term W_i as weighted degree of node i, or simply weight of node i) and off-diagonal entries l_{ij} such that l_{ij} is $-w_{ij}$ if (i, j) is an edge, and l_{ij} is 0 otherwise. That is

$$l_{ij} = \begin{cases} W_i & if \;\; i = j \\ -w_{ij} & if \;\; (i,j) \in E \\ 0 & otherwise \end{cases}$$

Observe that $L(G)$ is a symmetric matrix. The weighted Laplacian matrix can be obtained from the following equation: $L = D - W$, where D is a diagonal matrix with $d_{ii} = W_i$. The Laplacian matrix can also be written in terms of the incidence matrix U: $L = U \times Diag(Vec(W)) \times U^t$, where $Vec(W)$ is a m-by-1 vector obtained from concatenating the rows of matrix W. This equation can also be written in the following form:

$$L = \sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t \tag{A.1}$$

Example A.2.1 *In the weighted fish network (Fig. A.2), the weight matrix is given as follows.*

$$W = \begin{pmatrix} 0 & 10 & 3 & 0 & 0 & 0 & 0 \\ 10 & 0 & 5 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 6 & 1 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 8 & 1 & 0 \end{pmatrix}$$

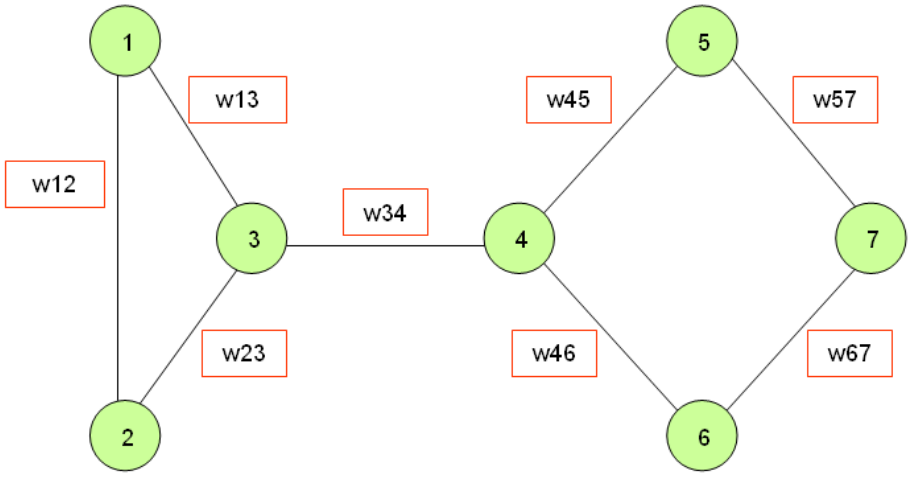


Figure A.2: Weighted Fish Network Topology

The Laplacian of weighted fish network is:

$$L = \begin{pmatrix} 13 & -10 & -3 & 0 & 0 & 0 & 0 \\ -10 & 15 & -5 & 0 & 0 & 0 & 0 \\ -3 & -5 & 15 & -7 & 0 & 0 & 0 \\ 0 & 0 & -7 & 14 & -6 & -1 & 0 \\ 0 & 0 & 0 & -6 & 14 & 0 & -8 \\ 0 & 0 & 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & 0 & 0 & -8 & -1 & 9 \end{pmatrix}$$

A.3 Eigenvalues of the Laplacian

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ denote the eigenvalues of L, with $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. For starters, observe that $\lambda_1 = 0$, corresponding to the eigenvector $(1, 1, \ldots, 1)^T$. The eigenvalue of interest to us is λ_2. It happens that $\lambda_2 = 0$ if and only if the graph G is disconnected. λ_2 is called "Algebraic Connectivity", and plays an important role in the theory of graphs.

From here on, assume G is connected. Let us call λ_2 the *eigenvalue of the graph*.

Let $u = (u_1, u_2, \ldots, u_n)^T$ be an eigenvector corresponding to λ_2. The eigenvectors of a symmetric matrix are orthogonal, so we have $u \perp (1, 1, \ldots, 1)^T$, so $\sum u_i = 0$.

A.3.1 The Rayleigh quotient

Let $x = (x_1, x_2, \ldots, x_n)^T$. Define the *Rayleigh quotient* to be

$$\phi_x = \frac{x^T L x}{x^T x} = \frac{\sum_{(i,j)\in E(G)} w_{ij}(x_i - x_j)^2}{\sum x_i^2}$$

Observe that

$$\lambda_1 = \min_{x \neq 0} \phi_x$$

and

$$\lambda_2 = \min_{x \perp (1,1,\ldots,1)^T} \phi_x.$$

A.4 Continuum vs. Combinatorial Differential Operators on Graphs

There is a simple correspondence between continuum differential operators and combinatorial differential operators on graphs, summarized in the following table.

Operator	Vector Calculus	Combinatorial
Gradient	∇	U
Divergence	$\nabla.$	U^t
Laplacian	$\nabla.\nabla = \nabla^2$	UU^t
Beltrami-Laplace	$\nabla W.\nabla$	$UDiag(Vec(W))U^t$

Table A.1: Continuum and Combinatorial Differential Operators on Graphs

Appendix B

Semi-Definite Programming

In the following we provide a short review of the semi-definite programming (SDP) method [97, 99, 100, 101].

B.1 Definition

A semi-definite program is the problem of minimizing a linear function of a variable $x = (x_1, x_2, ..., x_n)^t \in R^n$ subject to positive semi-definiteness of a certain matrix $F(x)$. The problem input consists of $c \in R^n$ and $n + 1$ symmetric matrices $F_0, F_1, F_2, ..., F_n \in R^{m \times m}$ and asks for

$$\begin{aligned} &Minimize \ < c, x > \\ &subject \ to \ \ F(x) \geq 0 \end{aligned} \tag{B.1}$$

where $< c, x >= c^t x$, $F(x) \geq 0$ means that $F(x)$ is a positive semi-definite matrix, and

$$F(x) = F_0 + \sum_{i=1}^{n} x_i F_i$$

Proposition B.1.1 *The feasible region of optimization problem B.1 is a convex set.*

Proof The feasible set of problem B.1 is $F = \{x \in R^n | F(x) \geq 0\}$. $F(x)$ is a positive semi-definite matrix, that is, $< F(x)z, z >$ for all $z \in R^m$. Suppose $x, y \in F$ and take $\lambda \in [0, 1]$.

Then for every $z \in R^m$, we have

$$< (\lambda F(x) + (1-\lambda)F(y))z, z >= \lambda < F(x)z, z > +(1-\lambda) < F(y)z, z >\geq 0$$

Therefore $\lambda F(x) + (1-\lambda)F(y) = F(\lambda x + (1-\lambda)y) \geq 0$ and hence $\lambda x + (1-\lambda)y \in F$.

∎

A semi-definite program is a convex optimization problem since its objective function is a convex function, and the set of feasible solutions is a convex set.

B.2 Some Examples of Semi-Definite Programs

In this section, through some examples, we show that many optimization problems, even complicated non-linear problems, can be represented as a semi-definite program.

Example B.2.1 *Consider the following linear program (LP).*

$$\begin{aligned} &\textit{Minimize} \quad < c, x > \\ &\textit{Subject to} \quad Ax + b \geq 0 \end{aligned} \tag{B.2}$$

where $c \in R^n$, $A \in R^{m \times n}$, and $b \in R^n$.

Inequality B.2 is a componentwise inequality since $Ax + b \in R^m$. Note that a vector $x \geq 0$ (componentwise) if and only if $diag(x) \geq 0$ (i.e., the diagonal matrix containing components of x on the diagonal is positive semi-definite). Let $a_1, a_2, ..., a_n$denote the columns of A. We have

$$diag(Ax + b) = \sum_{i=1}^{m} x_i diag(a_i) + diag(b)$$

Therefore, problem B.2 is a semi-definite program with $F(x) = F_0 + \sum_{i=1}^{n} x_i F_i$, where $F_0 = diag(b)$ and $F_i = diag(a_i)$ for $i = 1, 2, ..., n$.

Example B.2.1 shows that a semi-definite program can be considered as generalization of a linear program (LP). Let us now consider a non-linear optimization problem.

Example B.2.2 *Let $A_0, A_1, A_2, ..., A_n \in R^{m\times m}$ be symmetric matrices, and for $x \in R^n$ set $A(x) = A_0 + \sum_{i=1}^{n} x_i A_i$. Consider the following optimization problem.*

$$Minimize \ \ \lambda_{max}(A(x))$$

$$Subject\ to \ \ x \in U$$

where U is a linear subspace of R^n. The above problem can be translated into a semi-definite program by introducing an auxiliary variable $t \in R$.

$$Minimize \ \ t$$

$$Subject\ to \ \ tI - A(x) \geq 0$$

$$x \in U$$

The first constraint is equal to $t \geq \lambda_{max}(A(x))$. To see this, it is enough to note that a matrix A is positive semi-definite if and only if all of its eigenvalues are non-negative.

In many situations in semi-definite programs we need the following definition.

Definition B.2.3 *The* Schur *complement of a matrix of the form $\Theta = \begin{pmatrix} A & B \\ B^t & C \end{pmatrix}$ is: $A - BC^{-1}B^t$.*

Proposition B.2.4 *Matrix $\Theta = \begin{pmatrix} A & B \\ B^t & C \end{pmatrix}$ is positive definite (semi-definite) if and only if matrix C is positive definite and the Schur complement of Θ is positive definite (semi-definite).*

Proof See [93].

∎

Example B.2.5 *Consider the following optimization problem.*

$$Minimize \ \ \frac{(c^t x)^2}{d^t x}$$

$$Subject\ to \ \ Ax + b \geq 0$$

where we assume $d^t x > 0$ whenever $Ax + b \geq 0$. We introduce an auxiliary variable t that serves as an upper bound on the objective.

$$\begin{gathered} \textit{Minimize } t \\ \textit{Subject to } Ax + b \geq 0 \\ \frac{(c^t x)^2}{d^t x} \leq t \end{gathered}$$

The second constraint can be represented as a semi-definite constraint using proposition B.2.4. Consider matrix $\Omega = \begin{pmatrix} t & c^t x \\ c^t x & d^t x \end{pmatrix}$. The Schur complement of matrix Ω is equal to $t(d^t x) - (c^t x)^2$, but $t(d^t x) - (c^t x)^2 \geq 0$ (second constraint), therefore according to proposition B.2.4, $\Omega \succeq 0$. Finally, we can combine both constraint into a linear matrix inequality as follows.

$$\begin{gathered} \textit{Minimize } t \\ \textit{Subject to } \begin{pmatrix} diag(Ax + b) & 0 & 0 \\ 0 & t & c^t x \\ 0 & c^t x & d^t x \end{pmatrix} \succeq 0 \end{gathered}$$

This is clearly a semi-definite program.

Above examples show that many problems, including non-linear optimization problems, can be represented as a semi-definite program. So semi-definite programming offers a unified way to study the properties of and derive algorithms for a wide variety of convex optimization problems. Most importantly, however, semi-definite programs can be solved very efficiently both in theory and in practice. Standard methods such a barrier method, and interior point method [97] are developed to solve semi-definite programs. There are also some open-source and commercial products to solve semi-definite programming problems [102, 103].

Appendix C

Graph Products

In the following we provide the definition of two important graph products.

C.1 Kronecker (Tensor) Product of Two Graphs

Consider n-by-m matrix $A = [a_{ij}]$ and p-by-q matrix $B = [b_{ij}]$. The Kronecker product of matrices A and B is given by mp-by-nq matrix $C = A \otimes B$ as follows:

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ a_{n1}B & a_{n2}B & \dots & a_{nm}B \end{pmatrix}$$

Definition C.1.1 *We define a Kronecker product of two graphs as a Kronecker product of their adjacency matrices.*

Example C.1.2 *Fig. C.1 shows graph G_1 and its Kronecker product by itself $G_2 = G_1 \otimes G_1$*

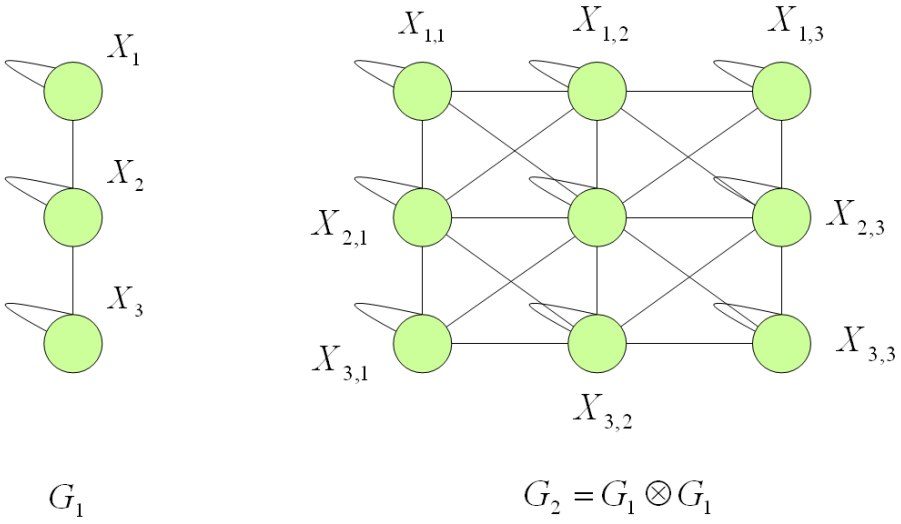


Figure C.1: Kronecker Product of Graph G_1 and Itself

[131]. In matrix form the adjacency matrix of graph G_1 is:

$$A(G_1) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

The adjacency matrix of G_2 according to the definition of Kronecker product is equal to:

$$A(G_2) = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

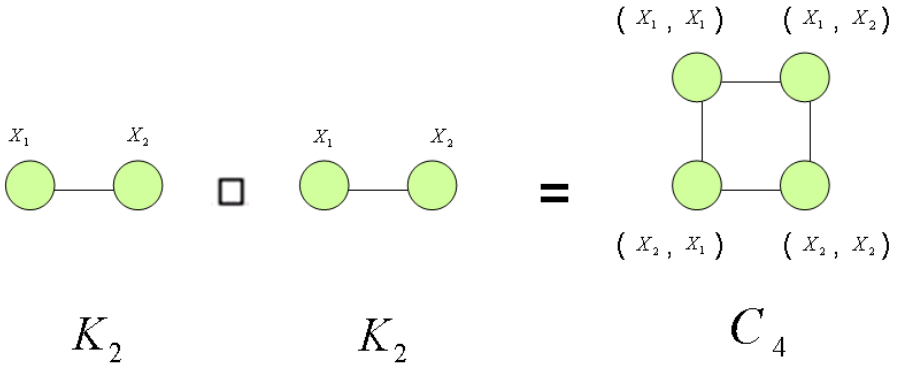


Figure C.2: Cartesian Product of 2 Simple Link Networks

C.2 Cartesian Product of Two Graphs

The cartesian product of two graphs $G(N_G, E_G)$ and $H(N_H, E_H)$ is given by $C = G\square H$, and defined as follows. The vertex set of $G\square H$ is the cartesian product of node sets of two graphs: $N_{G\square H} = N_G \times N_H$. Any two nodes (u, v) and (u', v') are adjacent (connected by a link) in $G\square H$ if and only if either

1. $v = v'$, and u and u' are adjacent in G.

2. $u = u'$, and v and v' are adjacent in H.

Example C.2.1 *Fig. C.2 shows that the cartesian product of two simple link networks (K_2) is a ring graph on 4 nodes (C_4).*

There is a simple relationship between cartesian and Kronecker product of graphs.

Proposition C.2.2 *Let H be a graph with adjacency matrix $A(G) \in R^{n\times n}$ and H be a graph with adjacency matrix $A(H) \in R^{m\times m}$. The cartesian product of G and H can be expressed in terms of the Kronecker product of G and H as follows.*

$$A(G\square H) = A(G) \otimes I_m + I_n \otimes A(H)$$

where I_m is the identity matrix of order m.

Proof See [93].

∎

C.3 Spectrum of Graph Products

Let the spectrum (set of eigenvalues) of graph G be $SP(G) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, and the spectrum of graph H be $SP(H) = \{\mu_1, \mu_2 \ldots, \mu_n\}$.

Proposition C.3.1 *The spectrum of the Kronecker product $G \otimes H$ is the following:*

$$SP(G \otimes H) = \{\lambda\mu \mid \ \lambda \in SP(G),\ \mu \in SP(H)\}$$

Furthermore, if x is the eigenvector of G associated with $\lambda \in SP(G)$ and y is the eigenvector of H associated with $\mu \in SP(H)$, then $x \otimes y$ is the eigenvector of $G \otimes H$ associated with $\lambda\mu$.

Proof See [93].

∎

Proposition C.3.2 *The spectrum of the cartesian product $G \square H$ is the following:*

$$SP(G \square H) = \{\lambda + \mu \mid \ \lambda \in SP(G),\ \mu \in SP(H)\}$$

Furthermore, if x is the eigenvector of G associated with $\lambda \in SP(G)$ and y is the eigenvector of H associated with $\mu \in SP(H)$, then $x \otimes y$ is the eigenvector of $G \square H$ associated with $\lambda + \mu$.

Proof This is a direct result of propositions C.2.2 and C.3.1.

∎

The smallest non-zero eigenvalue of the cartesian product of two graphs can be found as follows as a direct consequence of proposition C.3.2.

$$\lambda_2(G \square H) = \max\{\lambda_2(G),\ \lambda_2(H)\}$$

Finally, the largest eigenvalue of the cartesian product of two graphs is:

$$\lambda_{max}(G \square H) = \max\{\lambda_{max}(G),\ \lambda_{max}(H)\}$$

Appendix D

Sequence Diagrams for AutoNet

In this appendix we provide detailed diagrams for AVNM and ARM functionality. Fig. D.1 shows the block diagram of AVNM specifying list of functions of each sub-block.

Fig. D.2 shows the block diagram of ARM specifying list of functions of each sub-block.

Fig. D.3, D.4 and D.5 show a complete signal flow diagram including message exchanges between different blocks of AutoNet. These blocks are mainly the AVNM, ARM, network elements (NE), customer, and service provider.

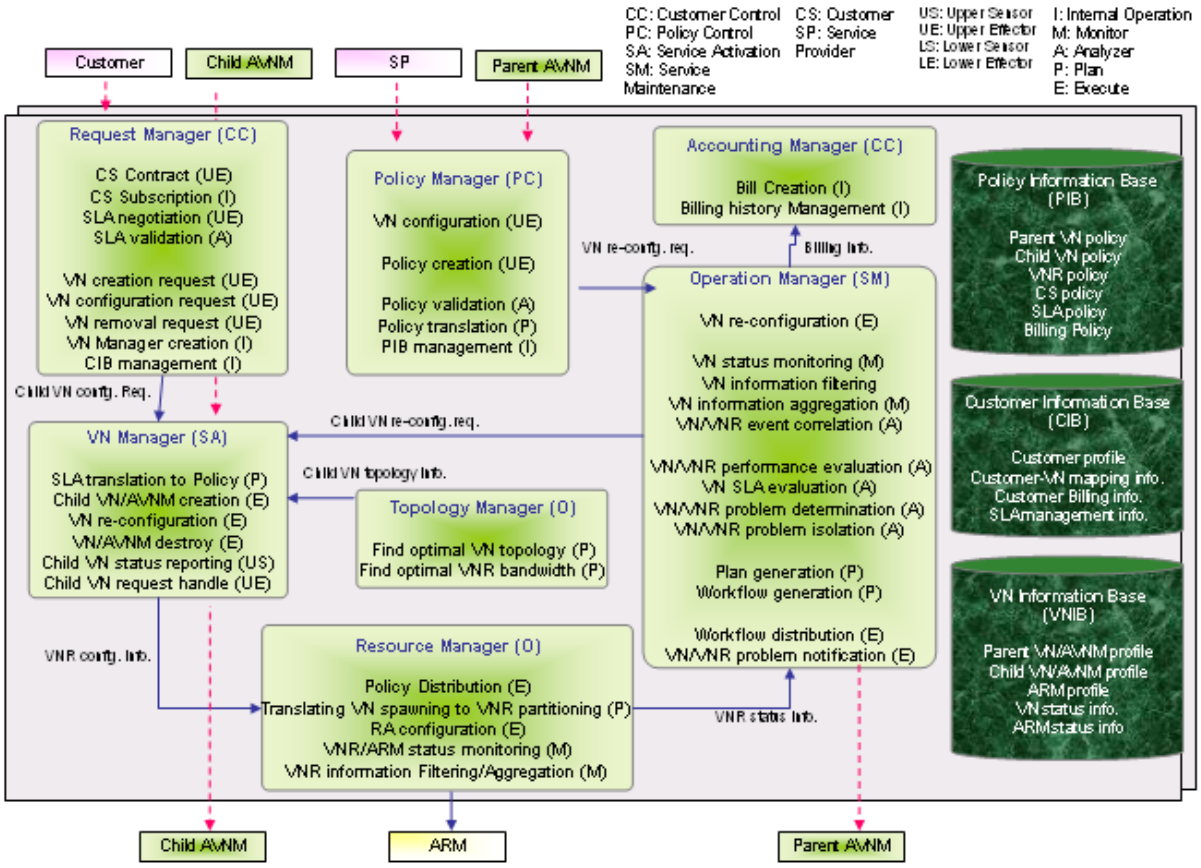


Figure D.1: Element-Wise Functionality of an AVNM

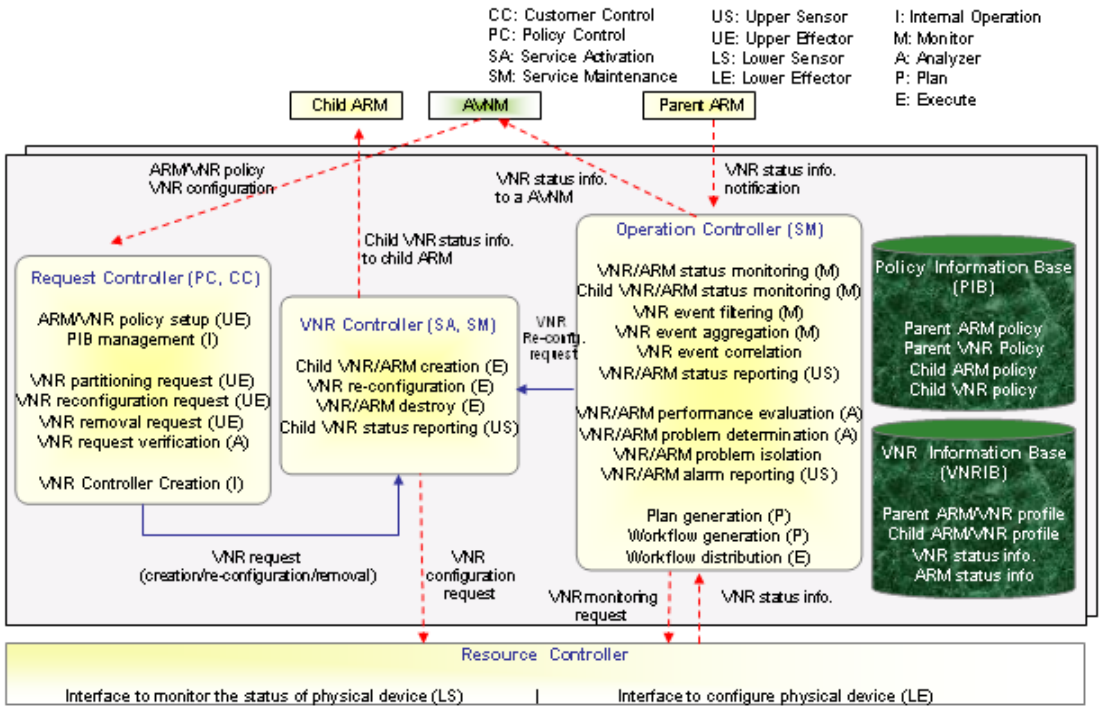


Figure D.2: Element-Wise Functionality of an ARM

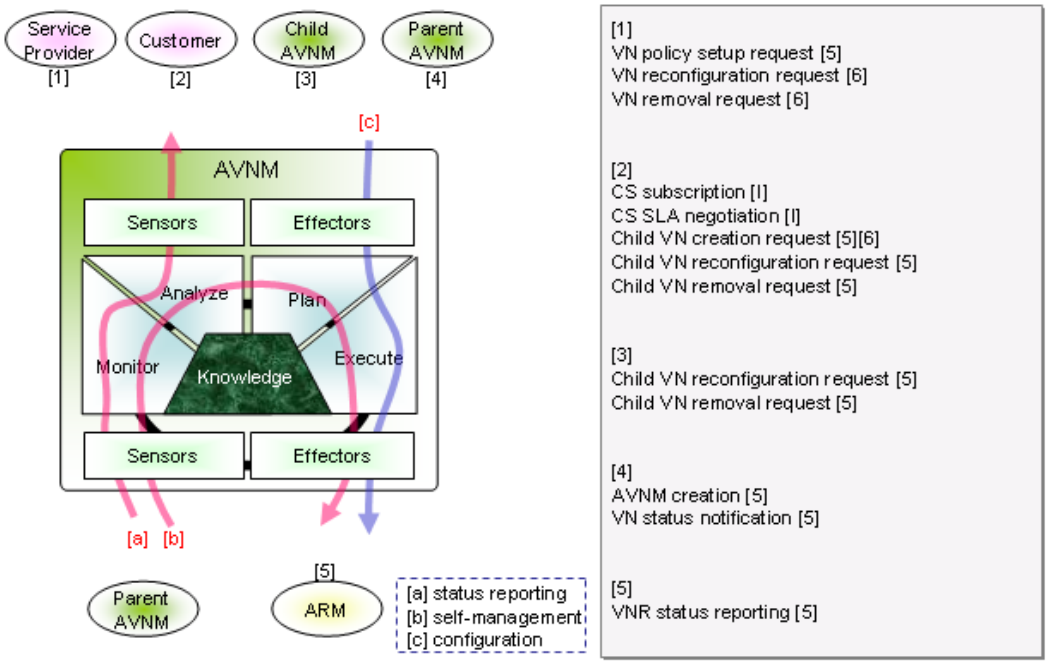


Figure D.3: Summary of Messages for AVNM

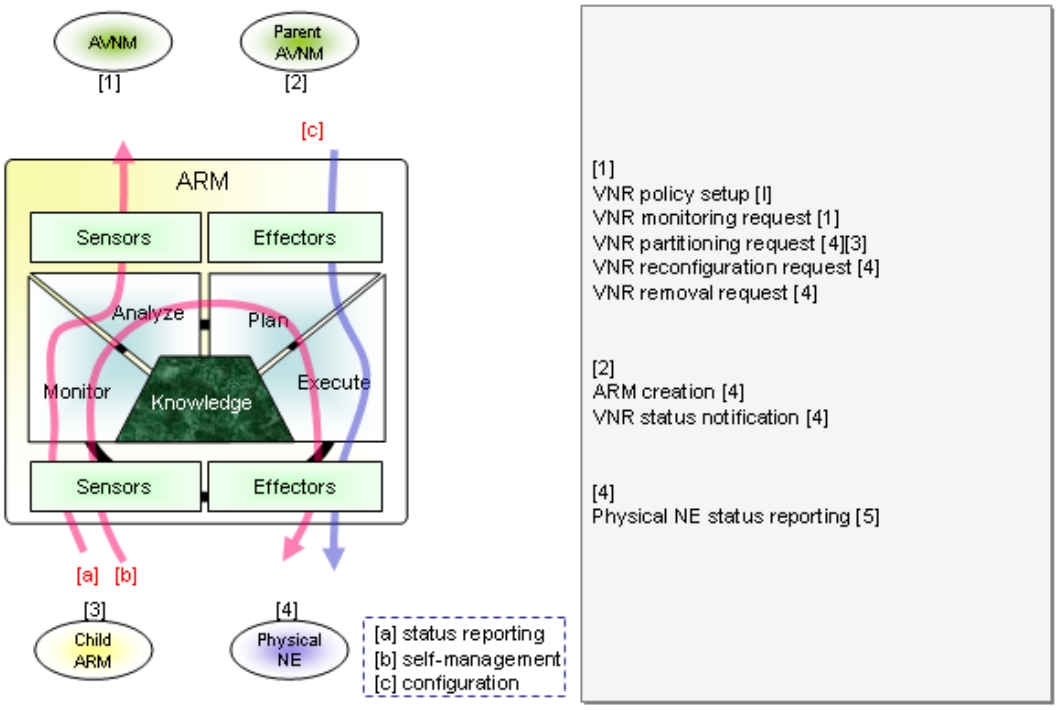


Figure D.4: Summary of Messages for ARM

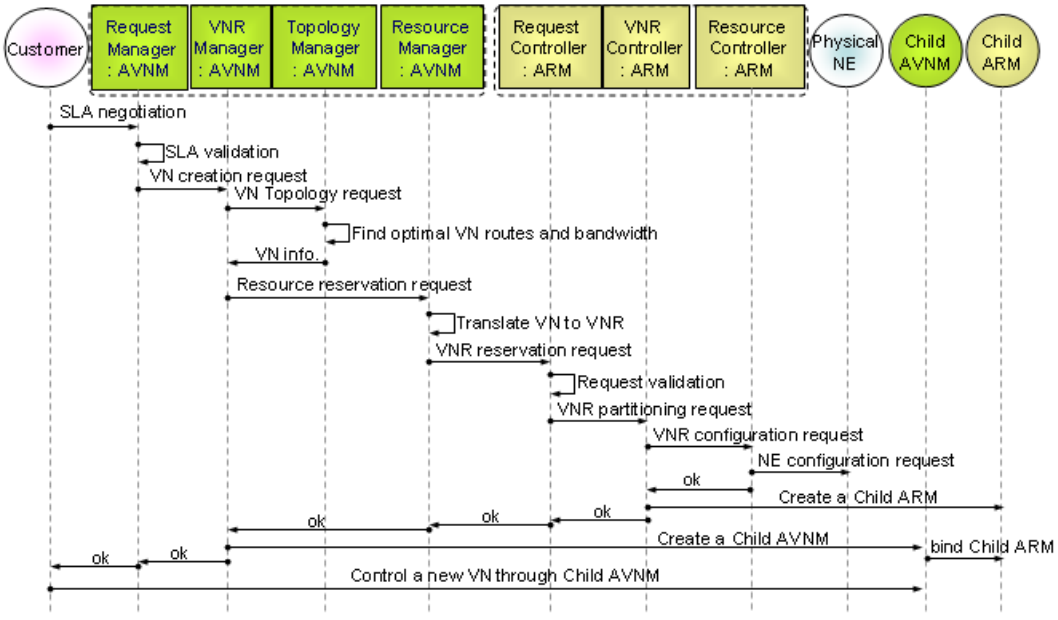


Figure D.5: Signal Flow for AVNM & ARM

Bibliography

[1] M. Nicolett, K. Brittain, and P. Adams. Enterprise Management ROI and Cost Reduction in 2003. Technical report, Gartner, November 2002.

[2] A. Tizghadam and A. Leon-Garcia. AORTA: Autonomic Network Control and Management System. In *Computer Communications Workshops, INFOCOM*. IEEE, 2008.

[3] M. S. Kim, A. Tizghadam, and A. Leon-Garcia. Virtual Network based Autonomic Network Resource Control and Management System. *IEEE GLOBECOM*, pages 1075–1079, 2005.

[4] Y. Cheng, R. Farha, A. Tizghadam, M. Kim, M. Hashemi, and A. Leon-Garcia. Virtual Network Approach to Scalable IP Service Deployment and Efficient Resource Management. *IEEE Communications Magazine*, 43(10):76–84, October 2005.

[5] A. Tizghadam and A. Leon-Garcia. LSP and Back up Path Setup in MPLS Networks Based on Path Criticality Index. *ICC 2007, IEEE*, pages 441–448, 2007.

[6] A. Tizghadam and A. Leon-Garcia. A Robust Routing Plan to Optimize Throughput in Core Networks. *ITC20, Elsvier*, pages 117–128, 2007.

[7] A. Tizghadam and A. Leon-Garcia. On Robust Traffic Engineering in Core Networks. In *IEEE GLOBECOM*, December 2008.

[8] A. Tizghadam and A. Leon-Garcia. On Congestion in Mission Critical Networks. In *Workshop on Mission Critical Networks (MCN), INFOCOM*. IEEE, April 2008.

[9] A. Tizghadam and A. Leon-Garcia. A Graph Theoretical Approach to Traffic Engineering and Network Control Problem. In *ITC21 (To be published)*, September 2009.

[10] A. Tizghadam and A. Leon-Garcia. On Autonomic Traffic Engineering. *Submitted to IEEE JSAC*, 2009.

[11] A. Tizghadam and A. Leon-Garcia. Survival Value of Communication Networks. *Infocom Workshop on Network Science for Communications (NetSciCom)*, April 2009.

[12] G. Leduca, H. Abrahamssone, S. Balona, S. Besslerb, M. D'Arienzoh, O. Delcourta, J. Domingo-Pascuald, S. Cerav-Erbasg, I. Gojmeracb, X. Masipd, A. Pescaph, B. Quoitinf, S.P. Romanoh, E. Salvadoric, F. Skivea, H.T. Tranb, S. Uhligf, and H. Umitg. An Open Source Traffic Engineering Toolbox. *Computer Communications*, 29(5):593–610, March 2006.

[13] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. Internet Engineering Task Force, RFC2748, May 2002.

[14] M. Rougan, M. Thorup, and Y. Zhang. Traffic Engineering with Estimated Traffic Matrices. In *Internet Measurement Conference*, pages 248–258, New York, NY, USA, 2003. ACM Press.

[15] M. Rougan, M. Thorup, and Y. Zhang. COPE: Traffic Engineering in Dynamic Networks. In *ACM SIGCOMM*, volume 36, pages 99–110, 2006.

[16] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Towsley. On Optimal Routing with Multiple Traffic Matrices. In *IEEE INFOCOM*, 2005.

[17] B. Fortz, J. Rexford, and M. Thorup. Traffic Engineering with Traditional IP Routing Protocols. *IEEE Communications Magazine*, (33), October 2002.

[18] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *INFOCOM*, pages 519–528, 2000.

[19] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for Interdomain Traffic Engineering. *ACM Computer Communication Review*, (33), October 2003.

[20] A.Sridharan, R. Guerin, and C. Diot. Achieving Near Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. *IEEE/ACM Transactions on Networking*, 13(2):234–247, 2005.

[21] B. Fortz and M. Thorup. Increasing Internet Capacity Using Local Search. *Computational Optimization and Applications*, 29(12):13–48, 2004.

[22] T.M. Thomas. *OSPF Network Design Solutions*. Cisco Press, 1998.

[23] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS Weights in a Changing World. *IEEE Journal in Selected Areas in Communications*, 20(4):756–767, 2002.

[24] K. Kar, M. Kodialam, and T. V. Lakshman. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *IEEE Journal on Selected Areas in Communications*, 18(12):2566–2579, Dec. 2000.

[25] S. Suri, M. Waldvogel, and P. R. Warkhede. *Profile-Based Routing: A New Framework for MPLS Traffic Engineering*, chapter Quality of Future Internet Services. Lecture Notes in Computer Science. Springle Verlag, September 2001.

[26] A. Elwalid, C. Jin, S. Low, , and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *Proceedings of IEEE INFOCOM'01*, pages 1300–1309, 2001.

[27] S. Yilmaz and I. Matta. On the Scalability-Performance Tradeoffs in MPLS and IP Routing. In *Proceedings of SPIE ITCOM*, May 2002.

[28] R. G. Gallager. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE Transactions on Communications*, 25:73–84, January 1977.

[29] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, and D. Gan. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC3209, December 2001.

[30] R. Guerin, D. Williams, and A. Orda. QoS Routing Mechanisms and OSPF Extensions. In *Proceedings of IEEE Globecom*, 1997.

[31] Z. Wang and J. Crowcroft. Quality of Service Routing for Supporting Multimedia Applications. *IEEE Journal in Selected Areas in Communications*, 14(7):1228–1234, 1996.

[32] J.C. de Oliveira, C. Scoglio, I.F. Akyildiz, and G. Uhl. A New Preemption Policy for Diffserv-Aware Traffic Engineering to Minimize Rerouting. In *Proceedings of IEEE Infocom*, June 2002.

[33] L. Melon, F. Blanchy, and G. Leduc. Decentralized Local Backup LSP Calculation with Efficient Bandwidth Sharing. In *Proceedings of IEEE ICT*, Papeete, Tahiti, February 2003.

[34] C. Scoglio, T. Anjali, J. Cavalcante, I. Akyildiz, and G. Uhl. TEAM: A Traffic Engineering Automated Manager for DiffServ-Based MPLS Networks. *IEEE Communications Magazine*, 42:134–145, October 2004.

[35] K. Kar and T.V. Lakshman M. Kodialam. Routing Restorable Bandwidth Guaranteed Connections Using Maximum 2-Route Flows. In *Proceedings of IEEE Infocom*, June 2002.

[36] K. Kar and T.V. Lakshman M. Kodialam. Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregated Link Usage Information. In *Proceedings of IEEE Infocom*, June 2001.

[37] S. Uhlig and B. Quoitin. Tweak-it: BGP-based Interdomain Traffic Engineering for Transit ASes. In *IEEE Conference on Next Generation Internet Networks Traffic Engineering*, Rome, Italy, April 2005.

[38] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain Traffic Engineering with BGP. *IEEE Communications Magazine*, May 2003.

[39] P. Aukia, M. Kodialam, P. V. N. Koppol, T. V. Lakshman, H. Sarin, and B. Suter. RATES: A Server for MPLS Traffic Engineering. *IEEE Network*, 14:34–41, March/April 2000.

[40] D. Durham, J. Boyle, R. Cohen, S. Herzog, and A. Sastry. The COPS (Common Open Policy Service) Protocol. Internet Engineering Task Force, RFC2748, January 2000.

[41] P. Trimintzios and et al. A Management and Control Architecture for Providing IP Differentiated Services in MPLS-Based Networks. *IEEE Communications Magazine*, 39(5):80–88, May 2001.

[42] N. G. Duffield, P. Goyal, and Albert Greenberg. A Flexible Model for Resource Management in Virtual Private Networks. *ACM SIGCOMM*, Aug. 1999.

[43] J. Bartlett. Optimizing Multi-Homed Connections. *Business Communications Review*, 32(1), 2002.

[44] D. Awduche, J. Agogbua, and J. McManus. An Approach to Optimal Peering Between Autonomous Systems in the Internet. In *Proceedings of International Conference on Computer Communications and Networks*, Lafayette, Louisiana, 1998.

[45] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Geographical and Temporal Characteristics of Inter-POP Flows: View From a Single POP. *European Transactions on Telecommunications*, 13(1):5–22, 2002.

[46] N. Feamster, J. Borkenhagen, and J. Rexford. Techniques for Inter-Domain Traffic Engineering. 2001 Technical Report 011106-02, AT&T Research, November 2001.

[47] N. Feamster and J. Rexford. Network-Wide BGP Route Prediction for Traffic Engineering. In *Proceedings of SPIE ITCOM Workshop on Scalability and Traffic Control in IP Networks*, August 2002.

[48] Cariden MATE. http://www.cariden.com/products/.

[49] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. NetScope: Traffic Engineering for IP Networks. *IEEE Network Magazine*, 14:11–19, 2000.

[50] I. Van Beijnum. *BGP: Building Reliable Networks with the Border Gateway Protocol*. OReilly and Associates, 2002.

[51] OPNET SP GURU. http://www.opnet.com.

[52] A. H. Dekker and B. D. Colbert. Network Robustness and Graph Topology. *Australasian Computer Science Conference*, 26:359–368, Jan. 2004.

[53] A. H. Dekker and B. Colberet. The Symmetry Ratio of a Network. In *Australasian symposium on Theory of computing*, volume 41, pages 13–20, Newcastle, Australia, 2005. ACM International Conference Proceeding Series.

[54] R. Zhang-Shen and N. McKeown. Designing a Predictable Internet Backbone with Valiant Load-Balancing. In *Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005.

[55] L. Valiant and G. Brebner. Universal Schemes for Parallel Communication. In *13th Annual Symposium on Theory of Computing*, May 1981.

[56] C. S. Chang, D. S. Lee, and Y. S. Jou. Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering. In *HPSR '01*, Dallas, May 2001. IEEE.

[57] I. Keslassy, S. T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown. Scaling Internet Routers Using Optics. In *SIGCOMM*, Karlsruhe, Germany, August 2003. ACM.

[58] M. C. Pinar, O. E. Karasan, and H. Yaman. The Robust Shortest Path Problem with Interval Data. Eprints for the Optimization Community, February 2003.

[59] R. Montemanni and L. M. Gambardella. An Exact Algorithm for the Robust Shortest Path Problem with Interval Data. *Computers and Operations Research*, 31(10):1667–1680, 2004.

[60] I. Averbakh. Minmax Regret Linear Resource Allocation Problems. *Operation Research Letters*, 32:174–180, 2004.

[61] A. Ben-Tal and A. Nemirovski. Lectures on Modern Convex Optimization: Analysis, Algorithms; Engineering Applications. *SIAM-MPS Series in Optimization*, 2000.

[62] A. Ben-Tal, A. Nemirovski, L. Ghaoui, and F. Rendl. *Handbook on Semidefinite Programming*. Kluwer Academic Publishers, 2000.

[63] D. Bertsimas and M. Sim. The Price of Robustness. *Operation Research*, 52(1):35–53, January-February 2004.

[64] P. Kouvelis and G. Yu. *Robust Discrete Optimisation and its Applications*. Kluwer Academic Publishers, 1997.

[65] H. Yaman, O. E. Karasan, and M. C. Pinar. The Robust Minimum Spanning Tree Problem with Interval Data. *Operations Research Letters*, 29:31–40, 2001.

[66] D. Applegate and E. Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: understanding fundamental tradeoffs. In *SIGCOMM*, pages 313–324, 2003.

[67] D. Applegate, L. Breslau, and E. Cohen. Coping with Network Failures: Routing Strategies for Optimal Demand Oblivious Restoration. In *SIGMETRICS*, pages 270–281, 2004.

[68] Y. Azar, E. Choen, A. Fiat, H. Kaplan, and H. Racke. Optimal Oblivious Routing in Polynomial Time. In *ACM Symposium in Parallelism in Algorithms and Architectures (SPAA)*, 2003.

[69] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Online Oblivious Routing. In *ACM SIGCOMM*, volume 36, pages 99–110, 2006.

[70] M. S. Kodialam, T. V. Lakshman, and S. Sengupta. Maximum Throughput Routing of Traffic in the Hose Model. In *INFOCOM*, pages 1217–1225. IEEE, 2006.

[71] Y. Li, J. Harms, and R. Holte. A Simple Method for Balancing Network Utilization and Quality of Routing. In *ICCCN*, San Diego,CA, 2005.

[72] S. Agarwal, C.-N. Chuah, S. Bhattacharyya, and C. Diot. The Impact of BGP Dynamics on Intra-Domain Traffic. In *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, New York, NY, June 2004.

[73] R. Teixeira, S. Agarwal, and J. Rexford. BGP Routing Changes: Merging Views from Two ISPs. *ACM SIGCOMM Computer Communications Review*, October 2005.

[74] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, New York, NY, June 2004. ACM.

[75] R. Teixeira, T. Griffin, A. Shaikh, and G. Voelker. Network Sensitivity to Hot-Potato Disruptions. In *ACM SIGCOMM*, Portland, OR, August 2004. ACM.

[76] J. Wu, Z. M. Mao, J. Rexford, and J. Wang. Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. In *USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 05)*, San Francisco, CA, May 2005.

[77] T. C. Bressoud, R. Rastogi, and M. A. Smith. Optimal Configuration for BGP Route Selection. In *INFOCOM*, New York, NY, June 2002. IEEE.

[78] N. Feamster, J. Winick, and J. Rexford. A Model of BGP Routing for Network Engineering. In *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, New York, NY, June 2004. ACM.

[79] S. Agarwal, A. Nucci, and S. Bhattacharyya. Measuring the Shared Fate of IGP Engineering and Interdomain Traffic. In *13th International Conference on Network Protocols (ICNP)*, Boston, MA, November 2005.

[80] M. Kodialam, T. V. Lakshman, and S. Sengupta. Traffic-Oblivious Network Routing For Guaranteed Bandwidth Performance. *IEEE Communications Magazine*, April 2007.

[81] C. Darwin. *The Origin of Species by Means of Natural Selection*. D. Appleton and Company, Available online at http://www.literature.org/authors/darwin-charles/the-origin-of-species/, 1859.

[82] IBM. An Architectural Blueprint For Autonomic Computing, April 2003.

[83] L. C. Freeman. Centrality in Networks: I. Conceptual Clarification. *Social Networks*, (1):215–239, 1978/79.

[84] L. S. P. Borgatti. Centrality and Network Flow. *Social Networks*, 27(1):55–71, 2005.

[85] D. Eppstein. Finding the K-Shortest Paths. *Society for Industrial and Applied Mathematics (SIAM) Journal of Computing*, 28(2):652–673, 1998.

[86] M. Newman. A Measure of Betweenness Centrality Based on Random Walks. *arXiv cond-mat/0309045.*, 2003.

[87] L. Kleinrock. *Queueing Systems*, volume II. John Wiley & Sons, 1975.

[88] P. Van Mieghem and F. A. Kuipers. Concepts of Exact QoS Routing Algorithms. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 12(5):851–864, October 2004.

[89] Michael William Newman. *The Laplacian Spectrum of Graphs*. PhD thesis, Department of Mathematics, University of Manitoba, July 2000.

[90] Fan R. K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series On Mathematics, No. 92. American Mathematical society, 1997.

[91] R. Grone, R. Merris, and V. S. Sunder. The Laplacian Spectrum of a Graph. *SIAM Journal, Matrix Analysis and Applications*, 11(2):218–238, April 1990.

[92] Jiayuan Huang. A Combinatorial View of Graph Laplacians. Technical Report 144, Max Planck Institute for Biological Cybernetics, 2005.

[93] Dennis S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005.

[94] M. Fiedler. Algebraic Connectivity of Graphs. *Czechoslovak Math. Journal*, 23(98):298–305, 1973.

[95] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. John Weily and Sons Inc., 1971.

[96] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, April 2003.

[97] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[98] R.H. Byrd, J. Nocedal, and R.A. Waltz. KNITRO: An Integrated Package for Nonlinear Optimization. *http : //www.ziena.com/papers/integratedpackage.pdf*. 2006.

[99] M. Todd. Semidefinite Optimization. *Acta Numerica*, 10:515–560, 2001.

[100] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook on Semidefinite Programming*. Kluwer, 2000.

[101] C. Helmberg. Semidefinite Programming for Combinatorial Optimization. *ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin*, October 2000.

[102] M. Grant and S. Boyd. CVX: Matlab Software for Disciplined Convex Programming (Web Page and Software). *http : //stanford.edu/ boyd/cvx*. September 2008.

[103] M. Grant and S. Boyd. Graph Implementations for Nonsmooth Convex Programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, *http : //stanford.edu/ boyd/graph$_d$cp.html*. *Lecture Notes in Control and Information Sciences, Springer*, 2008.

[104] L. Kleinrock. *Communication Nets, Stochastic Message Flow and Delay*. McGraw-Hill, New York, 1964.

[105] B. Meister, H. R. Muller, and H. R. Rudin. New optimization criteria for message switching networks. *IEEE Transactions on Communication Technology*, 19(3):256–260, June 1971.

[106] A. Leon-Garcia and L. Mason. Virtual Network Resource Management for Next-Generation Networks. *IEEE Communications Magazine*, 41(7):102–109, July 2003.

[107] A. Jun and et al. Virtual Network Resources Management: A Divide-and-Conquer Approach for the Control of Future Networks. *IEEE Globecom, Sydney Australia*, 2:1065–1070, November 1998.

[108] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *IEEE Computer Magazine*, 36(1):41–50, January 2003.

[109] K. Appleby, S. B. Calo, J. R. Giles, and K. W. Lee. Policy-Based Automated Provisioning. *IBM Systems Journal*, 43(1):121–135, 2004.

[110] D. Mitra and K.G.Ramakrishnan. A Case Study of Multiservice Multipriority Traffic Engineering Design for Data Networks. In *IEEE GLOBECOM*, volume 99, pages 1077–1083, December 1999.

[111] F. G. Chung. The Diameter and Laplacian Eigenvalues of Directed Graphs. *Electronic Journal of Combinatorics*, 13(4):1–19, April 2006.

[112] F. G. Chung. Laplacians and the Cheeger Inequality for Directed Graph. *Annals of Combinatorics*, 9(1):1–19, April 2005.

[113] C. W. Wu. Algebraic Connectivity of Directed Graphs. *Linear and Multilinear Algebra*, 53(3):203–223, June 2005.

[114] R. Agaev and P. Chebotarev. On the Spectra of Nonsymmetric Laplacian Matrices. *Linear Algebra and its Applications*, 399(1):157–168, April 2005.

[115] J. M. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14:95–116, 1984.

[116] A. Jamakovic and S. Uhlig. On the Relationship between the Algebraic Connectivity and Graphs Robustness to Node and Link Failures. In *3rd EuroNGI Conference, On Next Generation Internet Networks*, pages 96–102, Trondheim, May 2007.

[117] A. Jamakovic and P. Van Mieghem. On the Robustness of Complex Networks by Using the Algebraic Connectivity . In *NETWORKING 2008 Ad Hoc and Sensor*

Networks, Wireless Networks, Next Generation Internet, pages 183–194. Springer Berlin / Heidelberg, May 2008.

[118] J. E. Boillat. Load Balancing and Poisson Equation in a Graph. *Concurrency: Practice and Experience*, 2(4):289–313, December 1990.

[119] J. E. Boillat. Load Balancing Algorithms Based on Gradient Methods and their Analysis through Algebraic Graph Theory. *Journal of Parallel and Distributed Computing*, 68(2):209–220, February 2008.

[120] S. Winter. Route Specification with a Linear Dual Graph. In *Advances in Spatial Data Handling: Proc. 10th Int. Symp. Spatial Data Handling (SDH 2002)*, pages 329–338. Springer-Verlag, 2002.

[121] A. Basu, A. Lin, and S. Ramanathan. Routing using Potentials: A Dynamic Traffic-Aware Routing Algorithm. In *ACM SIGCOMM*, pages 27–38, 2003.

[122] D. M. Cvetkovic, M. Doob, and H. Sachs. *Spectra of Graphs: Theory and Applications*. Academic Press, 1980.

[123] S. Balon and G. Leduc. Dividing the traffic matrix to approach optimal traffic engineering. In *IEEE ICON*, 2006.

[124] G. Apostolopoulos. Using multiple topologies for IP-only protection against network failures: A routing performance perspective. Technical report, ICS-FORTH, Greece, 2006.

[125] S. Gjessing and O. Norway. Implementation of two resilience mechanisms using multi topology routing and stub routers. In *AICT-ICIW*, 2006.

[126] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast IP network recovery using multiple routing congurations. In *IEEE INFOCOM*, 2006.

[127] K. Kwong, R. Guerin, A. Shaikh, and S. Tao. Improving Service Differentiation in IP Networks through Dual Topology Routing. In *CoNEXT*, 2007.

[128] M. Dorigo and T. Sttzle. *Ant Colony Optimization*. MIT Press, July 2004.

[129] G.D. Cario and M. Dorigo. AntNet: Distributed stigmergetic control of communication networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[130] A. Tizghadam, M. Hashemi, and A. Leon-Garcia. Investigation of Ant Colony algorithms in Multiple Traffic Flow Environments. In *CCECE/CCGEI*, pages 1006–1009, Saskatoon, May 2005. IEEE.

[131] J. Leskovec and C. Faloutsos. Scalable Modeling of Real Graphs using Kronecker Multiplication. In 24^{th} *International Conference on Machine Learning, Wireless Networks, Next Generation Internet*, volume 227, pages 497 – 504, Corvalis, OR, 2007. ACM.